Savitribai Phule Pune University
(Formerly University of Pune)

F.Y.B.Sc.(Computer Science)

With

Major: Computer Science

(Faculty of Science and Technology)

(For Colleges Affiliated to Savitribai Phule Pune University)

Choice Based Credit System (CBCS) Syllabus Under
National Education Policy (NEP)
To be implemented from Academic Year 2024-2025

Title of the Course: B.Sc.(Computer Science)

F.Y.B.Sc.(Computer Science)

Semester-II

Lab Course – I

Work Book

Name:_____

College Name: _____

Roll No.: _____ **Division: _____**

Academic Year : _____

BOARD OF STUDIES

1. Dr. Patil Ranjeet
2. Dr. Joshi vinayak
3. Dr. Wani Vilas
4. Dr. Mulay Prashant
5. Dr. Sardesai Anjali
6. Dr. Shelar Madhukar
7. Dr. Bharambe Manisha
8. Dr. Deshpande Madhuri
9. Dr. Kardile Vilas
10. Dr. Korpal Ritambhara
11. Dr. Gangurde Rajendra
12. Dr. Manza Ramesh
13. Dr. Bachav Archana
14. Dr. Bhat Shridhar

Co-ordinators

- **Dr. Prashant Mulay**, Annasaheb Magar College, Hadapsar , Pune.
  Member, BOS Computer Science, Savitribai Phule Pune University
- **Dr. Sardesai Anjali** , Modern College of Arts, Science and Commerce , Shivajinagar, Pune
  Member, BOS Computer Science, Savitribai Phule Pune University

Editor

Dr. Prashant Mulay, Annasaheb Magar College, Hadapsar , Pune.
Member, BOS Computer Science, Savitribai Phule Pune University

Prepared by:

Ms. Gadekar Manisha Jankiram.	Annasaheb Magar College, Hadapsar, Pune.

Introduction

About the work book:

This LAB book / Workbook is intended to be used by F.Y.B.Sc. (Computer Science) students for the Advanced C Assignments in Semester–II. This workbook is designed by considering all the practical concepts / topics mentioned in syllabus. The lab book is to be used as a hands-on resource, reference and record of assignment submission and completion by the student. The lab book contains the set of assignments which the student must complete as a part of this course.

The objectives of this LAB-Book are:

Defining the scope of the course.

- To bring uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
- To have continuous assessment of the course and students.
- Providing ready reference for the students during practical implementation.
- Provide more options to students so that they can have good practice beforefacing the examination.
- Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

Instructions to the students

- Students are expected to carry this book every time they come to the lab for computer science practical.
- Students should prepare oneself beforehand for the Assignment byreading the relevant material.
- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor.
- However, student should spend additional hours in Lab and at home to coveras many problems as possible given in this work book.

Submission:

Problem Solving Assignments:

- The problem solving assignments are to be submitted by the student in the form of a journal containing individual assignment sheets.
- Each assignment includes the Assignment Title, Problem statement, Date of submission,
- Assessment date, Assessment grade and instructors sign.

Programming Assignments:

Programs should be done individually by the student in the respective login.

The codes should be uploaded on either the local server, Moodle, Github or any open source LMS. Print-outs of the programs and output may be taken but not mandatory for assessment.

Assessment:

Continuous assessment of laboratory work is to be done based on overall performance and lab assignments performance of student.

Each lab assignment assessment will be assigned grade/marks based on parameters with appropriate weightage.

Suggested parameters for overall assessment as well as each lab assignment assessment include- timely completion, performance, innovation, efficient codes and good programming practices.

Operating Environment:

For Advanced 'C' Programming:

Operating system: Linux

Editor: Any linux based editor

like vi, edit etc.

Compiler: cc or gcc.

Students will be assessed for each exercise on a scale from 0 to 5.

Not done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

Instruction to the Instructors

- Explain the assignment and related concepts in around ten minutes using whiteboard if required or by demonstrating the software.
- You should evaluate each assignment carried out by a student on ascale of 5 as specified above by ticking appropriate box.
- The value should also be entered on assignment completion page of the respective Lab course.

Lab Course I

Section II

Advanced “ C ” Programming

Assignment Completion Sheet

Lab Course I			
Sr. No	**Assignment Name**	**Marks**	**Sign**
1	**Pointers : Operations on pointers** • Pointers - Declaration, • Definition, initialization, • Dereferencing • Pointer arithmetic.		
2	**Pointers : Pointers and functions** • Passing pointer to function • Returning pointer from function, • Function pointer		
3	**Pointers : Pointers and arrays** • Pointer to array, • Array of pointers • Pointer to pointer		
4	**Pointers :Dynamic Memory allocation / Dangling pointers and Free** • malloc(), • calloc(), • Resizing(realloc()), • Releasing (free ())., • Dangling pointers		
5	**Strings : Basic operations / Array of strings** • String Literals, string variables, declaration, definition, initialization and Syntax and use of predefined string functions. • Array of strings		
6	**Strings and Pointer** • Strings with Pointers		
7	**Structures : Basics and Nested structure** • Structure, definition and initialization, use of typedef. • Accessing structure members and Nested Structures		
8	**Array of Structures and functions** • Arrays of Structures and functions- Passing		

	each member of structure as a separate argument, • Passing structure by value / address.		
9	**Pointers and Structures / Unions** • Use of Pointers and Structures • Concept of Union, declaration, definition, accessing union members.		
10	**Command Line Arguments** • To access command-line arguments • Functions - atoi(), atol() and atof() • Arithmetic operation on arguments • Accessing string using command line arguments.		
11	**File Handling** • Streams and Types of files. • Operations on text files. • Standard library input/output functions and Random access to files. • Accessing file using command line arguments.		
12	**Preprocessor** • Preprocessor and Format of preprocessor directive • File inclusion directives (#include) Macro substitution directive, argumented and nested macro and macros versus functions.		
	Total		

Name and Signature of Batch In-charge **Head of Department**

Date

Internal Examiner **External Examiner**

Assignment 1 : Pointers : Operations on pointers

Objective:
To useful for accessing memory locations
To understand pointer stores the address of another variable

Reading:
You should read following topics before starting this exercise

1. What is a pointer?
2. How to declare and initialize pointers.
3. '*' and '&' operators.
4. Pointer arithmetic operation.

Ready References:
Pointers
Pointers are variables that contain the address of another variable within the memory.
A Pointer is a reference to another variable (memory location) in a program.
Syntax :

data_type or return_type *pointer_name;

Example :

int *ptr_val;

ptr_val : Name if the pointer variable.
*(Asterisk) : Indicates that this variable is a pointer.
int : Data type of the pointers object.

float *ptr_val1;

declares that 'ptr_val1' is a pointer that points to floating point variable.

char *ptr_val2;

declares that 'ptr_val2' is a pointer that points to character point variable.

Example :

int a = 100;

The complier automatically assigns reference(memory) for this value. Every location in the memory has a unique address, we represent the a's location in the memory as below,

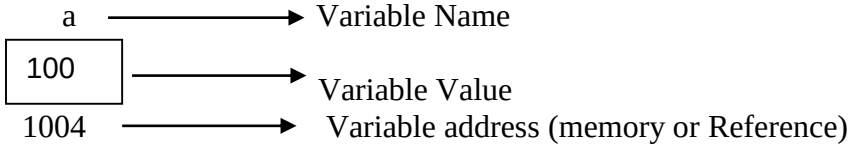
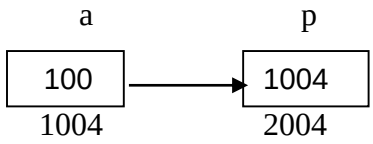


Fig : Representation of a in the memory

The variable a is associated with the memory address 1004. Since memory addresses are just numbers, they can also be assigned to any variables that can be stored in the memory location, like other variables. Such variables that contain the memory address are called Pointers.

```
int a = 100;
int *p;
p = &a;
```

We assign the address of a to variable p, then p is called the pointer as it points to the variable a.



Dereferencing Pointers :

The value of variable using pointer can be accessed using unary operator * (asterisk) , known as the **indirection operator** or **dereferencing operator.**

Example :

```
int p1,p2,*ptr ;
p1 = 77;
ptr = &p1;        // assigns the address of variable 'p1' to pointer 'ptr'.
p2 = *ptr;        // assigns the value at address pointed by 'ptr' to variable p2.
p2 = *(&p1);   // p2 = *ptr are equivalent to the  P2 = *(&p1).
p2 = p1;          // which is equivalent to.
```

/*Ex1. Write a program to illustrate the use if dereferencing or indirection oprator.*/

```
#include<stdio.h>
main()
{
      int p1 = 88 ,*ptr;
      ptr = &p1;
     printf("\n Value of p1 = %d",p1);
     printf("\n Address of p1 = %u",ptr);
     printf("\n Value of p1 = %d",*ptr);
     printf("\n Value of p1 = %d",*(&p1));
}
/*OUTPUT :
Value of p1 = 88
Address of p1 = 2008
Value of p1 = 88
Value of p1 = 88
*/
```

Pointer Arithmetic

Valid operations

- Pointer++, Pointer--
- Pointer + integer.
- Pointer – integer.
- Pointer – Pointer (must point to elements of same array).
- Comparison.
- Assignment.

Invalid operations

- Pointer + Pointer
- Pointer * Pointer
- Pointer / Pointer
- Pointer % Pointer

Consider int *ptr;

Operator	Expression	Meaning
Increment (++)	ptr ++;	Increments ptr to points next location of same type. If ptr is a pointer to integer (int occupies 4 bytes), and if ptr = 4002 then ptr++ gives ptr = 4006. If ptr is a pointer to character (char occupies 1 bytes), and if ptr = 4002 then ptr++ gives ptr = 4003.
Decrement (--)	ptr --;	Decrements ptr to points previous location of same type. If ptr is a pointer to integer (int occupies 4 bytes), and if ptr = 4006 then ptr-- gives ptr = 4002. If ptr is a pointer to character (char occupies 1 bytes), and if ptr = 4003 then ptr-- gives ptr = 4002.
Adding a number to pointer	ptr = ptr +5;	ptr points to 5 integer locations after current location. If ptr = 4002 the ptr+5 gives ptr = 4007
Subtracting a number from a pointer	ptr = ptr - 2;	ptr points to 2 integer locations before current location. If ptr = 4002 the ptr-2 gives ptr = 4000
Subtraction of one pointer from another pointer	ptr = ptr1 – ptr2;	We can subtract one pointer from another if and only if both pointing two one array. Here ptr,ptr1,ptr2 all are pointers to same array.

Pointers can be compared using relational operators. OR Pointer Comparison	ptr1 == ptr2;
	ptr1 < ptr2;
	ptr1 <= ptr2;
	ptr1 > ptr2;
	ptr1 >= ptr2;
	ptr1 != ptr2;
		Allowed.

/*Ex2. program illustrate how to handle integer values using pointers */

```
#include<stdio.h>
int main()
{
int a,b;
printf("\nEnter the numbers");
scanf("%d%d",&a,&b);
int *pa, *pb;
pa=&a;
pb=&b;
*pb=*pa+5;
printf("a=%d\tb=%d\n",a,b);
printf("*pa=%d \t *pb=%d\n",*pa,*pb);
return 0;
}
/*OUTPUT
Enter the numbers
25
45
a=25    b=30
*pa=25 *pb=30
*/
```

/*Ex3. Write a program Arithmetic operations using pointers. */

```
#include<stdio.h>
#include<string.h>
int main()
{
int y,x;
printf("Enter the value to perform arithmetic operations on it.\n");
scanf("%d",&x);
scanf("%d",&y);
int *p,*q;
p=&x;
q=&y;
int a,b,c,d;
a=(*p+*q);
b=(*p)-(*q);
c=(*q)*(*p);
d=(*p)/(*q);
printf("\nAddition = %d \nSubtraction = %d \nMultipication = %d \nDivide =
%d",a,b,c,d);
return 0;
```

```
}
/*OUTPUT
Enter the value to perform arithmetic operations on it.
500
200
Addition = 700
Subtraction = 300
Multipication = 100000
Divide = 2
*/
```

NULL Pointer

- A pointer that doesn't point to any memory locations is called a NULL pointer.
- The NULL constant is defined in the header files stdio.h, and stdlib.h.
- It is used to perform error handling with pointers before dereferencing the pointers.

Syntax

```
datatype  *PointerName = NULL;
```

Example

```
int *ptr = NULL;
int *prt = 0;
```

/*Ex4. Write a program using NULL pointers. */

```
#include <stdio.h>
int main()
{
  int *ptr = NULL;   // NULL pointer
  if (ptr == NULL)
  {
    printf("Pointer is a NULL ");
  }
  else
  {
    printf("If it is not then value stored in the address referred by the pointer :
            %d", *ptr);
  }
    return 0;
}
/*OUTPUT
Pointer is a NULL
*/
```

void Pointer :

- When we declare an int pointer, it can only point to int-type variables; it cannot point to float variables or any other kind of variable. We employ a pointer to void in order to get around this issue.
- A pointer to void means a generic pointer that can point to any data type.

Syntax

void *pointer name;

Example

void *ptr;

```
/*Ex4. Write a program using void pointers. */

#include <stdio.h>
int main()
{
  int a = 111;
  char b = 'M';
  void *ptr = &a;  // void pointer holds address of int a
  printf("\nValue of 'a': %d",a);
  printf("\nAddress of 'a': %d", &a);
  printf("\nVoid pointer points to: %d", ptr);
  ptr = &b;          // points to char b
  printf("\nValue of 'b': %c",b);
  printf("\nAddress of 'b': %d", &b);
  printf("\nVoid pointer points to: %d", ptr);
}
/*OUTPUT
Value of 'a': 111
Address of 'a': 416670388
Void pointer points to: 416670388
Value of 'b': M
Address of 'b': 416670387
Void pointer points to: 416670387*/
```

Set A : Write C programs for the following problems.

1. Write a program to accept an integer using pointer and check whether it is even or odd.
2. Write a program to find maximum from two integers using pointers.
3. Write a program to read two integers using pointers and perform all arithmetic operations on them.

4. Write a program to interchange value of two variables using pointer.
5. Write a program to sum of first ‘n’ numbers using pointers.

Signature of the Instructor | | Date | |

Set B : Write C programs for the following problems.

1. Write a program to accept radius and display area and perimeter using pointer.
2. Write a program to sum of ‘n’ numbers using pointers.
3. Write a program to display the reverse of the number using pointer.
4. Write a program to accept number is input through the keyboard, write a program to obtain the sum of the first and last digit of this number.

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 2 : Pointers :Pointers and functions

Objective:

To demonstrate use of pointers in C.

Reading:

You should read following topics before starting this exercise

1. Passing pointer to function,
2. Returning pointer from function,
3. Function pointer

Ready References:

Pointers and Functions

We can pass the address of a variable to a function. The function can accept this address in a pointer and use the pointer to access the variable's value.

Syntax

datatype function_name(type *var1, type *var2, ...)

Passing Pointers to Functions

It gets around pass by value's drawback. At the address included in the pointer, changes are made directly to the value inside the called function. We can therefore change the variables in one scope from another.

It also gets around a function's restriction of only returning one expression. When pointers are sent, a function's processing effect occurs right at the address.

/*Ex1. Write a program display the addition of two numbers using references of two variables*/

```
#include <stdio.h>
int add(int *, int *);     //function declaration
int main()
{
   int a,b;
   printf("\nEnter the numbers = ");
   scanf("%d%d",&a,&b);
   int sum = add(&a, &b);       //function declaration passing references of numbers
   printf("Addition: %d", sum);
   return 0;
}
int add(int *x, int *y)         //function definition
{
   int z = *x + *y;

   return z;
}
```

```
/* OUTPUT
Enter the numbers = 15   25
Addition: 40
*/
```

/*Ex2. Write a program display factorial of given number.*/

```
#include <stdio.h>
int fact(int *);     // function declaration
int main()
{
  int n;
  printf("\nEnter the number = ");
  scanf("%d",&n);
  int f = fact(&n);     //function declaration passing references of numbers
  printf("Factorial is = %d", f);
}
int fact(int *n) //function definition
{
  int f=1;
  for(int i=1;i<=*n;i++)
   {
     f = f * i;
   }
  return f;
}
/*OUTPUT
Enter the number = 6
Factorial is = 720
*/
```

Set A : Write programs to solve the following problems.

1. Write a function to swap value of two variables using pointer.
2. Write a function to accept the number and check that number is prime or not.
3. Write a function to accept the limit and display Fibonacci series upto the limit.
4. Write a function which takes hours, minutes and seconds as parameters and an integer ‘s’ and increments the time by ‘s’ seconds. Accept time and seconds in main and Display the new time in main.

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program to calculate the GCD of two numbers using pointer functions.
2. Write a function which display all armstrong numbers in between 1 to 500.
3. Write a function which accepts a number and three flags as parameters. If the number is even, set the first flag to 1. If the number is prime, set the second flag to1.If the number is divisible by 3or7, set the third flag to 1. In main, accept an integer and use this function to check if it iseven, prime and divisible by 3 or 7. (Hint : pass the addresses of flags to the function)
4. Write a function which takes distance in kilometer, centimeter and millimeter as parameters as an integer d and increments the distance by d millimeters. Accept distance and d in main and Display the new distance in main using the above function.

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 3 : Pointers : Pointers and Arrays

Objective:

To demonstrate use of array of pointers and pointer to pointer.

Reading:

You should read following topics before starting this exercise

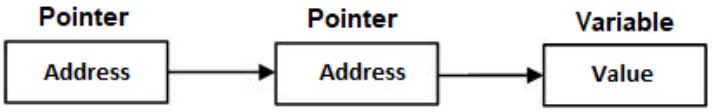
1. Relationship between array and pointer.
2. Pointer to array and Array of pointers.

Ready References:

Pointer To Pointer

Pointer is known as a double pointer (pointer to pointer).

The address of a variable is stored in the first pointer, and the address of the first pointer is stored in the second pointer.



Syntax

```
datatype **pointer_to_pointer;
```

Example

```
int a;
int * p;
int **q;
p = &a;
q = *p ;
```

/*Ex1. Program to display the variable value using pointer to pointer*/

```
#include <stdio.h>
int main()
{
   int val;
   printf("\nEnter the value = ");
   scanf("%d",&val);
   int *ptr = &val;// Pointer to integer
   int **dptr = &ptr; // Pointer to pointer ie double pointer
   printf("Value of 'val' is : %d\n", val);
   printf("Value of 'val' using pointer (ptr) is : %d\n", *ptr);
   printf("Value of 'val' using double pointer (dptr) is : %d\n", **dptr);
   return 0;
```

```
}
/*OUTPUT
Enter the value = 555
Value of 'val' is : 555
Value of 'val' using pointer (ptr) is : 555
Value of 'val' using double pointer (dptr) is : 555
*/
```

Pointer to array

An array name is a constant pointer to the first element of the array i.e., The memory address of the first element is the same as the name of the array.

Syntax

datatype (*var_name)[size]

Example

int (*p)[5];

Array of Pointers

An array name is a pointer to the first element in the array. It holdsthe base address of the array.

Syntax

datatype *array_name[size];

Example

int *p[5];

Every array expression is converted to pointer

For example :

- a[i] is same as*(a+i)
- a[i][j] is same as *(*a+i)+j)
- &a[i] is same as a+i
- &a[i][j] is same as a[i]+j

int n[10];

- *n and * (n + 0) : represents0th element
- n[j], *(n+ j),* (j + n) , j[n] : represent the value of the jth element of array n.

```
/*Ex2. Program to read and display array elements*/

#include<stdio.h>
int main()
{
   int arr[10];
   int *ptr = arr;
   int i,n;
   printf("\nEnter the limit = ");
   scanf("%d",&n);
   printf("\nEnter the array Element\n");
   for (i = 0; i < n; i++)
   {
    scanf("%d", &arr[i]);
   }
   printf("\nArray Element\n");
```

```
    for (i = 0; i < n; i++)
    {
      printf("%d ", *(ptr + i));
    }
return 0;
}
/*OUTPUT
Enter the limit = 4
Enter the array Element
11
22
33
44
Array Element
11 22 33 44
*/
```

```
/*Ex3. Program to read and display array elements using function pointer */

#include<stdio.h>
void arr_display(int *ptr,int n);      //function prototype
int main()
{
    int arr[10];
    int *ptr = arr;
    int i,n;
    printf("\nEnter the limit = ");
    scanf("%d",&n);
    printf("\nEnter the array Element\n");
    for (i = 0; i < n; i++)
    {
     scanf("%d", &arr[i]);
    }
    arr_display(arr,n);          //passing array
    return 0;
}
   void arr_display(int *ptr,int n)    //function defination
   {
    int i;
    printf("\nArray Element\n");
    for (i = 0; i < n; i++)
    {
      printf("%d ", *(ptr + i));    //accessing array elements
    }
  }
/*OUTPUT
Enter the limit = 5
Enter the array Element
```

```
1
4
7
9
11
Array Element
1 4 7 9 11
```

```
/*Ex4. Program to display 2D array element using pointer */

#include<stdio.h>
int main()
{
   int i,j;
   int a[2][2]={{10, 5}, {18, 25}};    //constant array declaration
   printf("\nMatrix are\n") ;
   for(i=0; i<2;i++)
   {
      for(j=0; j<2; j++)
     {
        printf("%d ", *(*(a+i)+j));     //retrieving array elements
      }
    printf("\n");
  }
  return 0;
}
/*OUTPUT
Matrix are
10 5
18 25
*/
```

Set A : Write programs to solve the following problems.

1. Write a program to accept n numbers in an array and display elements sum of array elements using pointer.
2. Write a program to accept n numbers calculate the range of elements in the array using pointer.
3. Write a function to accept n numbers and display the array in the reverse order. Write separate functions to accept and display using pointer.
4. Write a function to accept n numbers and store all prime numbers in an array called prime.Display this array using pointer.
5. Write a program to accept a matrix A of size mxn and display the element also display the maximum elements of matrix.

6. Write a program to accept a matrix A of size mxn and store its transpose in matrix B. Display transpose matrix use pointer.

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program to find the union and intersection of the two sets of integers use array and function pointer also write a separate function for each.
2. Write function to accept n numbers and print Array Elements Present in Even Position and display the sum of elements.
3. Write a function to accept n numbers and print the Sum of Even and the Product of Odd Digits use array pointer.
4. Write a function to accept n numbers and Delete an Element from an Array and Print a New Array use pointer.
5. Write a program to accept n numbers sort the elements using arrange in descending order using function and pointer.
6. Write a function for Linear Search, which accepts an array of n elements, store them in an array. Accept the key to be searched and search it using this function. Display appropriate messages use pointer.
7. Write a menu driven program to perform the following operations on a square matrix. Writeseparate functions with pointer for each option.
    i. Display the trace of the matrix (sum of diagonal elements).
    ii. Check if the matrix is an upper triangular matrix.
    iii. Check if the matrix is a lower triangular matrix.
    iv. Exit

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 4 : Pointer - Dynamic Memory allocation / Dangling pointers and Free

Objective:
To demonstrate advanced use of pointers

Reading:
You should read following topics before starting this exercise

1. Dynamic memory allocation (malloc, calloc, realloc, free).
2. Dangling pointers

Ready References:

Dynamic Memory allocation

Memory can be allocated for data variables after the program begins execution.
This mechanism is known as runtime memory allocation or dynamic memory allocation**.**
To allocate memory dynamically library functions are malloc(), calloc(), realloc() and free() are used.
Functions are defined in the <stdlib.h> header file.

Dynamic Memory allocation function

malloc()

"malloc" or "memory allocation" method is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.It initializes each block with default garbage value.

Syntax

```
ptr = (cast-type*) malloc(byte-size)
```

Example:

```
ptr = (int*) malloc(100 * sizeof(int));
```

calloc()

"calloc" or "contiguous allocation" method in C is used to dynamically allocate the specified number of blocks of memory of the specified type.It initializes each block with a default value '0'.

Syntax

```
ptr = (cast-type*)calloc(n, element-size);
```

Example

```
ptr = (float*) calloc(25, sizeof(float));
```

realloc()

"realloc" or "re-allocation" function in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to dynamically re-allocate memory.Re-allocation of memory maintains the already present value and new blocks will be initialized with default garbage value.

Syntax

```
ptr = realloc(ptr, newSize);
```

where ptr is reallocated with new size 'newSize'.

free()

"free" function in C is used to dynamically de-allocate the memory. The memory allocated using functions malloc() and calloc() is not de-allocated on their own.Hence the free() function is used, whenever the dynamic memory allocation takes place.It helps to reduce wastage of memory by freeing it.

Syntax

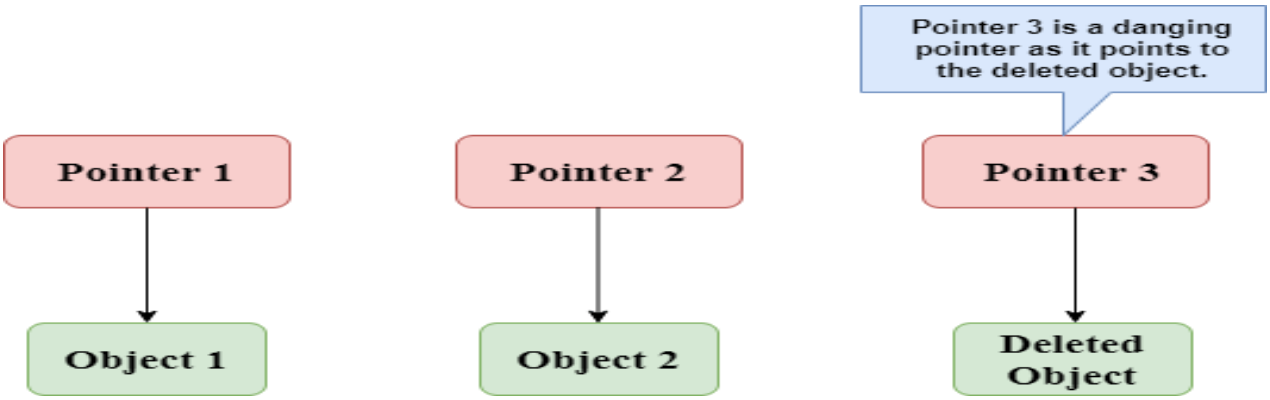
```
free(ptr);
```

Dangling Pointers

The programmer fails to initialize the pointer with a valid address, then this type of initialized pointer is known as a dangling pointer. Dangling pointer occurs at the time of the object destruction when the object is deleted or de-allocated from memory without modifying the value of the pointer. The pointer is pointing to the memory, which is de-allocated.

If we assign the value to this pointer, then it overwrites the value of the program code or operating system instructions; in such cases, the program will show the undesirable result or may even crash. If the memory is re-allocated to some other process, then we dereference the dangling pointer will cause the segmentation faults.

Example



In the above figure, we can observe that the Pointer 3 is a dangling pointer. Pointer 1 and Pointer 2 are the pointers that point to the allocated objects, i.e., Object 1 and Object 2, respectively. Pointer 3 is a dangling pointer as it points to the de-allocated object.

/*Ex1. Program to read and display array elements using dynamic memory allocation */

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int n,i,*p;
   printf("Enter number of elements: ");
   scanf("%d",&n);
   p=(int*)malloc(n * sizeof(int));  //memory allocated using malloc
   if(p == NULL)
   {
      printf("memory cannot be allocated\n");
   }
   else
   {
     printf("Enter elements of array:\n");
     for(i=0;i<n;++i)
     {
       scanf("%d",&*(p+i));
     }
     printf("Elements of array are\n");
     for(i=0;i<n;i++)
     {
       printf("\na[%d] = %d",i,*(p+i));
     }
      printf("\nArray are\n");
     for(i=0;i<n;i++)
     {
       printf("%d ",*(p+i));
     }
   }
   return 0;
}
/*OUTPUT
Enter number of elements: 5
Enter elements of array:
2
5
7
9
11
Elements of array are
a[0] = 2
a[1] = 5
```

```
a[2] = 7
a[3] = 9
a[4] = 11
Array are
2 5 7 9 11
*/
```

/*Ex2. Program to read and display array elements using dynamic memory allocation */

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int *a[10],r,c,i,j;
printf("Enter the row and column size of matrix\n");
scanf("%d%d",&r,&c);
printf("Enter matrix elements\n");
for(i=0;i<r;i++)
{
/**** dynamically allocate memory for every row ****/
a[i]=(int *)malloc(c*sizeof(int));
for(j=0;j<c;j++)
{
  scanf("%d",a[i]+j);
}
}
/****** Display Matrix ******/
printf("The matrix is as below\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",*(*(a+i)+j));
}
 printf("\n");
}
}
/*OUTPUT
Enter the row and column size of matrix
2
3
Enter matrix elements
1
2
3
```

```
4
5
6
The matrix is as below
1       2       3
4       5       6
*/
```

/*Ex3. Sample program of dangling pointer*/

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int a;
  printf("\nEnter the number = ");
  scanf("%d",&a);
  printf("\na = %d",a);
  int *ptr=(int *)malloc(sizeof(int));
  ptr=&a;
  free(ptr);
  printf("\nptr = %d",ptr);
  return 0;
}
```

/*Ex4. Sample program of dangling pointer*/

```
#include <stdio.h>
int *demo()
{
   int y=10;
   return &y;
}
int main()
{
int *p=demo();
printf("%d", *p);
return 0;
}
```

Set A : Write programs to solve the following problems.

1. Write a program to accept N integers and store them dynamically display them inreverse order.
2. Write a program to allocate memory dynamically for n integers such that the memory isinitializedto0. And display it.
3. Accept N integers in an array using dynamic memory allocation. Find maximum from them and display.
4. Accept a matrix of order m X n using dynamic memory allocation. Construct new matrix of order m X (n+1)(Use realloc function)such that $(n+1)^{th}$ column contains the sum of all elements of the corresponding row. And display new matrix.
   Example:

A			B			
10	20	33	10	20	3	**33**
4	15	6	4	15	6	**25**

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Accept n integers in an array. Copy only the non-zero elements to another array (allocated using dynamic memory allocation). Calculate the sum and average of non-zero elements.
2. Accept matrix of order mXn (Use dynamic memory allocation and array of pointers concept). Display trace of the matrix.
3. Write a program to accept two matrices , write separate functions to accept, display, the matrices. Accept a matrices of order mXn (Use dynamic memory allocation and array of pointers) do the following operations on ;

   i. Addition
   ii. Subtraction
   iii. Multiplication
   iv. Division

4. There are 5 students numbered 1 to 5. Each student appears for different number of subjectsin an exam. Accept the number of subjects for each student and then accept

the marks for each subject. For each student, calculate the percentage and display. (Hint: Use array of 5 pointers and use dynamic memory allocation)

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 5 : Strings : Basic operations and Array of strings

Objective:

To demonstrate strings.

Reading:

You should read following topics before starting this exercise

1. String literals
2. Declaration and definition of string variables
3. The NULL character
4. Accepting and displaying strings
5. String handling functions

Ready References:

String

A string is an array of characters terminated by a special character called NULL character(\0). Each character is stored in 1 byte as its ASCII code.

An array of strings is a two dimensional array of characters. It can be treated as a 1-D array such that each array element is a string.

Actions Involving strings	Explanation	Example
Declaring Strings	char string_name[size];	char str[80];
Initializing Strings		char str[]= { 'G', 'o', 'o',d','\0' }; char str [ ] = "Hello";

Accepting Strings	scanf and gets can be used to accept strings	char name[20], address[50];
		printf("\n Enter yourname");
		scanf("%s",name);
		printf("\n Enter your address");
		gets(address);

Displaying Strings	printf and puts can be used to display strings.	Printf("\n The name is %s:",name);
		printf("\n The address is :");
		puts(address);

Declaring String array	char array[size1][size2];	char cities[4][10]
Initializing String array		char cities[4][10] = { "Pune", "Mumbai","Delhi", "Chennai"} ;

All string operations are performed using functions in “string.h”. Some of the most commonly used functions are

String Function	Description
strlen(str)	Returns the number of characters in the string (excluding \0)
strcpy(deststr, srcstr)	Copies one string to another
strcmp(str1,str2)	Compares two strings. Returns 0 (equal), +ve (first string >second), -ve (first string <second). It is case sensitive.
strrev	Reverses a stringand returns the reversed string.
strcmpi(str1,str2)	Same as strcmp but ignores case
strcat(str1,str2)	Concatenates the second string to the first. Returns the concatenatedstring.
strchr(str1, ch)	Truncate the string before first occurrence of character in the string.
strrchr(str1,ch)	Truncate the string before last occurrence of character in the string.
strlwr(str)	Converts a string to lowercase.
strupr(str)	Converts a string to uppercase.

/*Ex1. Write a program use scanf() to read a string. */

```
#include <stdio.h>
int main()
{
   char name[20];
   printf("Enter name: ");
   scanf("%s", name);
   printf("Your name is %s.", name);
   return 0;
}
/*Output
```

```
Enter name: Dennis Ritchie
Your name is Dennis.
*/
```

/* Ex2. Write a program use gets() and puts().*/

```
#include <stdio.h>
int main()
{
   char name[30];
   printf("Enter name: ");
   gets(name);  // read string
   printf("Name: ");
   puts(name);    // display string
   return 0;
}
/*Output
Enter name: Dennis Ritchie
Your name is Dennis Ritchie
*/
```

/* Ex3. Write a program to use library function of string strlen()*/

```
#include <stdio.h>
#include <string.h>
int main()
{
   char a[20]="Program";
   char b[20]={'P','r','o','g','r','a','m','\0'};
   printf("Length of string a = %d",strlen(a));
   printf("Length of string b = %d",strlen(b));
   return 0;
}
/*Output
Length of string a = 7
Length of string b = 7
*/
```

/*Ex4. Program to find length of the string without using library string(ie. Strlen()) function
***/**

```
#include <stdio.h>
int main()
{
char str[100];
int i,len=0;
printf("Enter a string:");
```

```
gets(str);
for(i = 0; str[i]!= '\0'; ++i)
{
   len++;
}
printf("Length of string: %d",len);
return 0;
}
/*Output
Enter a string:happy diwali
Length of string: 12
*/
```

```
/*Ex5. Concatenation of two string using user defined function*/

#include<stdio.h>
void demostrcat(char str1[50], char str2[50]);     // function Prototype
int main()
{
 char str1[50], str2[50];
 int i, len=0;
 printf("Enter first string:\n");
 gets(str1);
 printf("Enter second string:\n");
 gets(str2);
 demostrcat(str1, str2);              //function declaration
 printf("Concatenated string is: %s", str1);
 return 0;
}
void demostrcat(char str1[50], char str2[50])      //function definatioin
{
 int i, len=0;
  for(i=0;str1[i]!='\0';i++)   // Calculating length of first string
 {
  len++;
 }
  for(i=0;str2[i]!='\0';i++)   //Concatenating second string to first string
 {
  str1[len+i] = str2[i];
 }
 str1[len+i]='\0';
}
/*Output
Enter first string:
Rock
Enter second string:
```

```
FYBSc(CS)
Concatenated string is: RockFYBSc(CS)
*/
```

Set A : Write programs to solve the following problems.

1. Write a program to read a string and copy it to another sting and display copied string. Also the length of copied string. (Use built in functions)
2. Write a program to read two strings. If first string is greater than second then concatenate second to first and display concatenated string, If first string is smaller than second then concatenate first to second, other-wise display length of string. (use strcmp)
3. Write a menu driven program to perform the following operations on string.
    i. Reverse of string
    ii. Convert to uppercase
    iii. Convert to lowercase
4. Write a program to read a string and one character. Check whether given character is present in the given string or not? (Hint: use strchr or strrchr)
5. Write a program to read a string and count total number of digits, alphabets and special characters.

Signature of the Instructor _____ Date _____

Set B : Write programs to solve the following problems.

1. Write a menu driven program to perform the following operations on two strings using without using library or built_in string function.
    i. Copy
    ii. Concatenation
    iii. Length of string
    iv. Compare
2. Write a program that accepts n words and outputs them in dictionary order. (Use array of pointers and dynamically allocate memory for each word)
3. Write a function that takes a string as parameter and returns the same string in uppercase. Accept this string in main and display converted string in main only.
4. Write a function to compare two strings. Write another function to reverse the string. In main function a string and check whether it is palindrome or not using above functions. (Hint: A palindrome string is a string which reads same in forward as well as backward direction for example: madam, nitin, etc.)

5. Write a program that will accept a string and character to search. The program will call a function, which will search for the occurrence of the character in the string and return its position. Function should return –1 if the character is not found in the string. Display this position in main() function.

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 6 : Strings and Pointers

Objective:

To demonstrate String operations using pointers.

Reading:

You should read following topics before starting this exercise

1. How to create and access an array of strings.
2. Dynamic memory allocation

Ready References:

Pointers to a string's first character are known as string pointers. Programmers can alter strings with these potent string manipulation tools without completely replicating them.

Declaring Pointer Variables for Strings

```
char *str = "AWESOME";
```

Accessing string using pointer

```
/*Ex1. Program to display the string using pointer*/

#include <stdio.h>
int main()
{
   char *str = " AWESOME ";
   printf("Hello !");
   while(*str != '\0')
   {
      printf("%c", *str);
      str++;                  // access the next character
   }
   return 0;
}
/*OUTPUT
Hello ! AWESOME
*/
```

```
/*Ex2. Program to display the string  and length using pointer*/

#include<stdio.h>
int main()
{
   char subjects[5][20];        // creating a pointer variable of size 5
                                // to store names of 5 different subjects
```

```
   int i, j,n;
   printf("Enter different subjects\n");
   scanf("%d",&n);
   printf("\nEnter the Subject \n");
   for(i = 0; i < n; i++)
   {
      scanf("%s", subjects[i]);
   }
      printf("The name of subjects are \n");
          for(i = 0; i < n; i++)  // accessing subjects
          {
              j = 0;          // initializing j = 0 to indicate first character
                               // of the subject at index i
          while (*(subjects[i] + j) != '\0')
          {
           printf("%c", *(subjects[i] + j)); // jth character of the string at index i is
                                              //*(subjects[i] + j)
           j++;
          }
             printf(" <-> size = %d\n", j); // it indicates length of the string
      }
   return 0;
}

/*OUTPUT
Enter different subjects
3
Enter the Subject
Computer
Electronics
Maths
The name of subjects and lengths are
Computer <-> size = 8
Electronics <-> size = 11
Maths <-> size = 5
*/
```

/* Ex3. Program to Compare Two Strings without using library function */

```
#include <stdio.h>
#include <string.h>
int Compare_Strings(char *Str1, char *Str2);   //function prototype
int main()
{
        char Str1[100], Str2[100];
        int result;
        printf("\nEnter the First String :  ");
        gets(Str1);
        printf("\nEnter the Second String :  ");
        gets(Str2);
        result = Compare_Strings(Str1, Str2); //function declaration
        if(result < 0)
        {
                printf("\n str1 is Less than str2");
        }
        else if(result > 0)
        {
                printf("\n str2 is Less than str1");
        }
        else
        {
                printf("\n str1 is Equal to str2");
        }
  return 0;
}
int Compare_Strings(char *Str1, char *Str2)  //function defination
{
        int i = 0;
        while(Str1[i] == Str2[i])
        {
                if(Str1[i] == '\0' && Str2[i] == '\0')
                        break;
                i++;
        }
        return Str1[i] - Str2[i];
```

```
}
/*OUTPUT
Enter the First String :  INDIA
Enter the Second String :  INDIA
 str1 is Equal to str2
*/
```

Set A : Write programs to solve the following problems.

1. Write a menu driven program to perform the following operations on two strings using pointer and write separate function for each operation.
   i. Length of string
   ii. Copy
   iii. Concatenation
   iv. Compare

2. Write a menu driven program to perform the following operations on one string using pointers.
   iv. Reverse of string
   v. Convert to uppercase
   vi. Convert to lowercase

   (Write a separate function for each operation)
3. Write a program which accepts a sentence from the user and alters it as follows: Every space is replaced by *, case of all alphabets is reversed, digits are replaced by ?.
4. Write a program that accepts n words and outputs them in dictionary order.
5. Define two constant arrays of strings, one containing country names (ex: India, France etc.,) and the other containing their capitals. (ex: Delhi, Paris etc.,). Note that country names and capital names have a one-one correspondence. Accept a country name from the user and display itscapital. Example: Input: India , Output: Delhi.

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a function which returns the substring of a given string using pointers. Use this function inmain. Function prototype: char * substring(char *str, int start, int end);
2. Write a function to compare two strings. Write another function to reverse the string. In main function a string and check whether it is palindrome or not using above functions. (Hint: A palindrome string is a string which reads same in forward as well as backward direction for example: madam, nitin, etc.)

3. Write a program that will accept a string and character to search. The program will call a function, which will search for the occurrence of the character in the string and return itsposition. Function should return –1 if the character is not found in the string. Display thisposition in main() function.
4. Write a program that will accept a string and character to search. The program will call a function, which will find occurrence of given character in the string. Display this count in main() function.
5. Write a program that accepts a string and generates all its permutations. For example: ABC,ACB, BAC, BCA, CAB, CBA

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 7 : Structures : Basics and Nested structure

Objective:

To understand execution of Structures.

Reading:

You should read following topics before starting this exercise

1. Concept of structure
2. Declaring a structure
3. Accessing structure members
4. Concept of nested structure

Ready References:

Structure

- Structure is a user-defined data type that enables us to store the collection of different data types. Each element of a structure is called a member.
- Structures can use as store various information
- The **struct** keyword is used to define the structure.

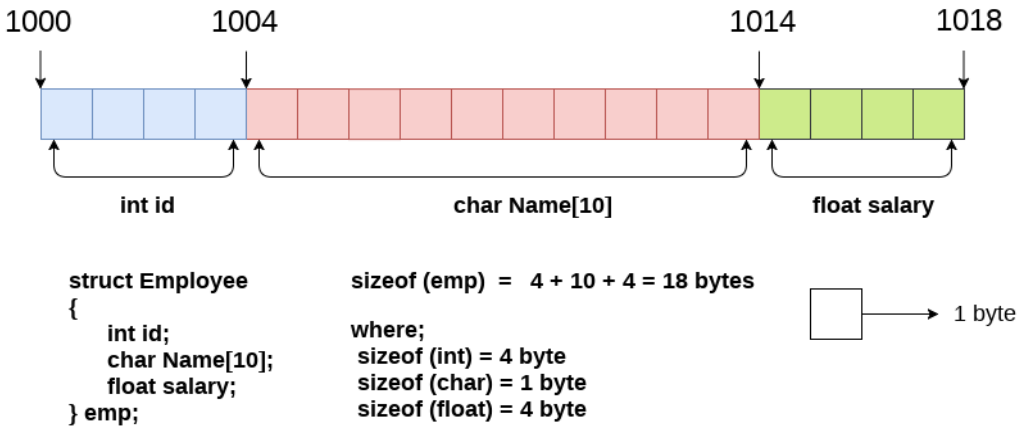
Syntax :

```
struct structure_name
{
        data_type member1;
        data_type member2;
         .
         .
        data_type memeberN;
};
```

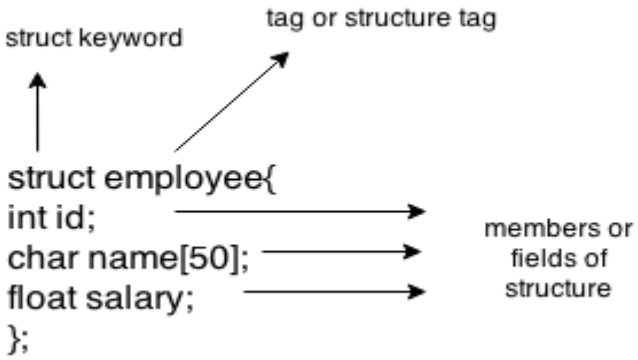
Example :

To define a structure for an entity employee

```
struct employee
{
          int id;
          char name[10];
          float salary;
};
```



Here, struct is the keyword; employee is the name of the structure; id, name, and salary are the members or fields of the structure.



Declaring structure Object

There are three ways to declare structure variable:

- By struct keyword within main() function
- By declaring a object at the time of defining the structure.
- By declaring a object after the structure definition and before main() function.

1:To declare Object within main() function

```
struct employee
{
   int id;
   char name[50];
   float salary;
};
 Now write given code inside the main() function.
int main()
{
      struct employee e1,e2;     //declaring the object e1 and e2
      …….
      ……..
}
```

2 :To declare Object at the time of defining the structure.

```
struct employee
{
   int id;
   char name[50];
   float salary;
}e1,e2;                 //declaring the object e1 and e2
```

3:To declare Object after structure declaration and before main() function

```
struct employee
{
   int id;
   char name[50];
   float salary;
};
      struct employee e1,e2;     //declaring the object e1 and e2
int main()
{
      …….
      ……..
}
```

/*Ex1. Write a program to declared and access the member of structure .*/

```
#include<stdio.h>
#include <string.h>
struct employee      //declaration of structure
{
    int id;                  //member declaration
   char name[50];
}e1;  //declaring e1 variable for structure
int main( )
{
     e1.id=101;   //store first employee information
     strcpy(e1.name, "FY");//copying string into char array
     printf( "employee 1 id : %d\n", e1.id);                  //printing first employee information
     printf( "employee 1 name : %s\n", e1.name);
return 0;
}
```

Output:

```
employee 1 id : 101
employee 1 name : FY
```

typedef Keyword

- The **typedef** is a keyword used in C programming to provide some meaningful names to the already existing variable in the program.
- It behaves similarly as we define the alias for the commands.
- In short, we can say that this keyword is used to redefine the name of an already existing variable.

Syntax of typedef

typedef <existing_name> <alias_name>

where, '**existing_name**' is the name of an already existing variable while '**alias name**' is another name given to the existing variable.

Example

```
/*Ex2. Write a program to declared and access the member of structure using typedef .*/

#include <stdio.h>
typedef struct customer
{
char cname[20];
int cage;
}cust;
int main()
{
cust c1;
printf("Enter the details of customer\n");
printf("\nEnter the name of the customer:");
scanf("%s",&c1.name);
printf("\nEnter the age of customer:");
scanf("%d",&c1.age);
printf("\n Name of the customer is : %s", c1.name);
printf("\n Age of the customer is : %d", c1.age);
return 0;
}
/*OUTPUT
Enter the details of customer
Enter the name of the customer: Rajveer
Enter the age of customer:21
Name of the customer is : Rajveer
Age of the customer is : 21
*/
```

Nested structure

Nested structure is a structure inside a structure. We refer to a nested structure when one of the items in the definition of one struct type is of another struct type.

When one of a struct type's elements is a composite representation of one or more types, this is known as a nested structure.

struct address_info

```
{
  int streetno;
  char streetname;
  int houseno;
};
struct student_ info
{
  char sname[50];
  int sid;
  struct address_ info address;      //nested structure
};
```

/*Ex3. Write a program to declared and access the member of nested structure .*/

```
#include <stdio.h>
struct address_info
{
  int street;
  char *state;
  char *city;
};

struct student_ info
{
  int sid;
  char *sname;
  struct address_info address; //nested structure
};
int main()
{
  struct student_info s1;
  s1.sid = 16;
  s1.sname = "Atharva";
  s1.address.street = 111;
  s1.address.state = "Maharashtra";
  s1.address.city = "Pune";

  printf("**********Student Data********** ");
  printf("\nStudent id: %d",s1.sid);
  printf("\nStudent name: %s",s1.sname);
  printf("\nStudent street: %d",  s1.address.street);
  printf("\nStudent state: %s",s1.address.state);
  printf("\nStudent city: %s",s1.address.city);
   return 0;
}
/*OUTPUT
```

```
**********Student Data**********
Student id: 16
Student name: Atharva
Student street: 111
Student state: Maharashtra
Student city: Pune
*/
```

Set A : Write programs to solve the following problems.

1. Write a program to create a structure person(pid, pname, page, psalary) read and display the information.
2. Write a program to create a structure customer(cno,cname,caddress) read and display the information of three customers.
3. Write a program to create a structure employee (id, name, salary) and nested structure DOB(date,month,year) read and display the information of two employees.
4. Write a program to create a structure hospital ,use the typedef to display the details.

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program to create a structure student (roll number, name, marks of 6 subjects, total, percentage). Accept details of 5 students and display the same.
2. Write a program to create a structure teacher(tno,tname,taddress) and structure experience(designation, joining_date) read and display the information of three teacher.
3. Write a program to create a structure person(pid, pname, page, psalary,pcontactno) read and display the information of two person use typedef.

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 8 : Array of Structures and functions

Objective:

To understand execution of array object of structures and functions.

Reading:

You should read following topics before starting this exercise

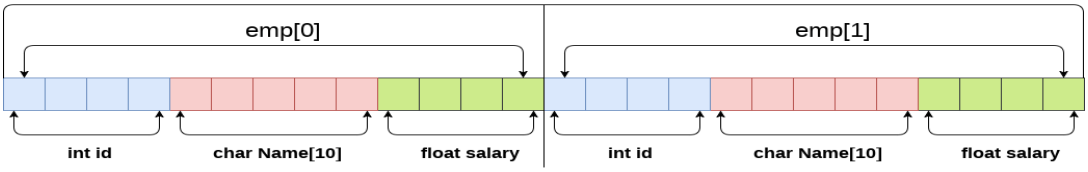
1. Accessing structure members
2. Array of structures
3. Passing structures to functions

Ready References:

Array of Structures

A collection of several structure variables, each containing information about a distinct entity, is known as an array of structures. Information about several entities of various data kinds can be stored using array structures. The collection of structures is another name for the array of structures.

Array of structures




```
struct employee
{
    int id;
    char name[5];
    float salary;
};
struct employee emp[2];
```

sizeof (emp) = 4 + 5 + 4 = 13 bytes

sizeof (emp[2]) = 26 bytes

/*Ex1. Write a program to read and display record of 'n' student using array of object. */

```
#include<stdio.h>
#include <string.h>
struct student
{
int rollno;
char name[10];
};
int main()
{
int i;
```

```
struct student st[5];
printf("Enter Records of 5 students");
for(i=0;i<5;i++)
{
printf("\nEnter Rollno:");
scanf("%d",&st[i].rollno);
printf("\nEnter Name:");
scanf("%s",&st[i].name);
}
printf("\nStudent Information List:");
for(i=0;i<5;i++)
{
   printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);
}
  return 0;
}
```

Passing structure to function

A variable that is used of the struct data type can be supplied to a function in the same way that we can pass arguments of primary data types. Additionally, call by value and call by reference methods can be used to pass structures.

/*Ex2. To pass the structure object employee to a function display() which is used to display the details of an employee*/

```
#include<stdio.h>
struct address
{
   char city[20];
   int pin;
   char phone[14];
};
struct employee
{
   char name[20];
   struct address add;
};
void display(struct employee);
void main ()
{
   struct employee emp;
   printf("Enter employee information?\n");
   printf("\nEnter the name,city,pin,phoneno\n");
   scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);
   display(emp);
```

```
}
void display(struct employee emp)
{
  printf("Employee Details....\n");
  printf("\nEmp Name\tEmp City\tEmp Pin\tEmp Contact\n");
  printf("%s \t%s %d %s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
/*OUTPUT
Enter employee information?
Enter the name,city,pin,phoneno
Veera
goa
123
8822555577
Employee Details....
Emp Name    Emp City    Emp Pin    Emp Contact
Veera       goa         123        8822555577
*/
```

Set A : Write programs to solve the following problems.

1. Write a program to create a structure person(pid, pname, page, psalary,pcontactno) read and display the information for 'n' number of person.
2. Write a program to create a structure hospital(hno,hname,haddress) accept and display the 'n' record of hospital use separate function of the same.
3. Write a program to create a structure student (roll number, name, marks of 3 subjects, percentage). Accept detailsof n students and write a menu driven program to perform the following operations. Write separate functions for the different options.
    i. Search by student number
    ii. Update the student address
    iii. Display all student details
    iv. Display all student having percentage > __
    v. Display student having maximum percentage

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program to create a structure employee (eid, ename, esalary). Accept details of n employees and write a menu driven program to perform the following operations. Write separate functions for the different options.
    i. Search by name
    ii. Search by id

iii. Display all
iv. Display all employees having salary > ____
v. Display employee having maximum salary

2. Write a program read name, salary and age of 'n' different customer as the three members of a structure named 'customer. Display the name, salary and corresponding age of the customer sorted on the basis of age in descending order.

3. Write a program to create a structure car (car_number, car_modelname, car_color, car_cost). Accept details of n cars and write a menu driven program to perform the following operations. Write separate functions for the different options.
   i. Search by name
   ii. Delete the car record
   iii. Display only 'red' color car
   iv. Display all

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 9 : Pointers and Structures / Unions

Objective:

Use of Pointers and Structures and Concept of Union

Reading:

You should read following topics before starting this exercise

1. Dynamic memory allocation
2. Creating and accessing unions

Ready References:

Structure Pointer

The structure pointer points to the address of a memory block where the structure is being stored.

Syntax :

Declare a structure pointer

```
struct structure_name *ptr;
```

Here structure pointer which tells the address of a structure in memory by pointing pointer variable ptr to the structure variable.

Initialization of the Structure Pointer

```
ptr = &structure_variable;
```

```
struct structureName
{
  datatype structure_member1;
  datatype structure_member2;

};
int main()
{
        struct structure_name *ptr;
        return 0;
}
```

Access Structure member using pointer:

There are two ways to access the member of the structure using Structure pointer:

- Using (*) asterisk or indirection operator and dot (.) operator.
- Using arrow (->) operator or membership operator.

```
pointer_name -> member_name;
```

```
/*Ex1. Write a program to accessing structure members using the dot operator*/
#include<stdio.h>
struct coordinate                    // create a structure Coordinate

{
      int x,y;   // declare structure members

};
int main()
{
   int p,q;
   struct coordinate first_point;
   // declaring structure pointer
   struct coordinate *cp;
   cp = &first_point;
   printf("\nEnter the coordinates\n");
   scanf("%d%d",&p,&q);

   (*cp).x = p;
   (*cp).y = q;

   printf("First coordinate (x, y) = (%d, %d)", (*cp).x, (*cp).y);
   return 0;
}
/*OUTPUT
Enter the coordinates
14
24
First coordinate (x, y) = (14, 24)
*/
```

```
/*Ex2. Write a program Accessing structure members using the arrow operator*/

#include<stdio.h>
struct Student
{
   char sname[30];
   int sage;
   int roll_number;
};
int main()
{
   struct Student stud1;
   struct Student *sptr = &stud1;   //pointer object
```

```
    printf ("Enter the details of the Student \n");
    printf ("\nEnter the Roll Number = ");
    scanf ("%d", & sptr ->roll_number);
    printf ("\nEnter the Student Name: ");
    scanf ("%s", sptr ->sname);
    printf ("\nEnter theSstudent Age: ");
    scanf ("%d", & sptr ->sage);

    printf ("\n Display the details of the student using Structure Pointer\n");
    printf ("Roll Number: %d\n", sptr->roll_number);
    printf ("Name: %s\n", sptr ->sname);
    printf ("Age: %d\n", sptr ->sage);
    return 0;
}
/*OUTPUT
Enter the details of the Student
Enter the Roll Number = 9
Enter the Student Name: Dua
Enter theSstudent Age: 20
Display the details of the student using Structure Pointer
Roll Number: 9
Name: Dua
Age: 20
*/
```

Union

Union can be defined as a user-defined data type which is a collection of different variables of different data types in the same memory location. The union can also be defined as many members, but only one member can contain a value at a particular point in time.

The keyword **union** is used to define unions. Union is a user-defined data type, but unlike structures, they share the same memory location. It means that the union members share the same memory location.

Syntax

```
union unionName
{
  member definition;
  member definition;
  ...
  member definition;
} one or more union variables/object;
```

Access the members of a union

```
union_name.member_name;
```

Initialization of Union Members

Initialize the members of the union by assigning the value to them using the assignment (=) operator.

Syntax

union_variable.member_name = value;

```
/*Ex3. Write a program to display data using union */

#include<stdio.h>
union demo
{
   int a;
   char b;
}obj;
int main()
{
  obj.a = 55;
  printf("\n a = %d", obj.a);
  printf("\n b = %d", obj.b);
  return 0;
}
/*OUTPUT
a = 55
b = 55
*/
```

```
/*Ex4. Write a program to display book information using union */
#include <stdio.h>
#include<stdlib.h>
#include <string.h>

union book
{
   int bno;
   float bprice;
   char bname[20];
};

int main()
{
   union book b1;
   printf("\nEnter the book infomation\n");
   printf("\nEnter the book number\n");
   scanf("%d",&b1.bno);
   printf("\nEnter the book name\n");
   scanf("%s",b1.bname);
```

```
   printf("\nEnter the book price\n");
   scanf("%f",&b1.bprice);
      printf("\nBook Number =  %d", b1.bno);
   printf("\nBook Name =  %s", b1.bname);
   printf("\nBook Price =  %f", b1.bprice);
   return 0;
}
/*OUTPUT
Enter the book infomation
Enter the book number
1
Enter the book name
Let Us C
Enter the book price
509.60
Book Number =  1
Book Name =  Let Us C
Book Price =  509.600006
*/
```

/*Ex5. Write a program to display book information using structure and union */

```
#include<stdio.h>
#include<stdlib.h>
struct library_book
{
int id;
char title[80],publisher[20] ;
int code;
union u
{
int no_of_copies; char month[10];
int edition;
}info;
int cost;
};
void main( )
{
struct library_book book1;
printf("\n Enter the details of the book \n");
printf("\n Enter the id, title and publisher \n");
scanf("%d%s%s",&book1.id, book1.title, book1.publisher);
printf("\nEnterthecode:1-TextBook,2-Magazine,3-Reference");
scanf("%d",book1.code);
switch(book1.code)
{
```

```
case 1:printf("Enter the number of copies :");
      scanf("%d",&book1.info.no_of_copies);
      break;
case 2:printf("Enter the issue month name:");
      scanf("%s",book1.info.month);
      break;
case 3:printf("Enter the editionnumber:");
         scanf("%d",&book1.info.edition);
         break;
}
printf("Enter the cost :");
scanf("%d",&book1.cost);
/* Display details of book */
printf("\n id = %d", book1.id);
printf("\n Title = %s", book1.title);
printf("\n Publisher = %s", book1.publisher);
switch(book1.code)
 {
    case 1:printf("Copies = %d:",book1.info.no_of_copies);
           break;
    case 2: printf("Issue month name =%s",book1.info.month);
           break;
    case 3:printf("Edition number=%d:",book1.info.edition);
           break;
  }
}
```

Set A : Write programs to solve the following problems.

1. Write a program to accept details for n books and write a menu driven program for the following:
    i. Display all reference books
    ii. Search magazine according for specific month
    iii. Find the total cost of all books (Hint: Use no_of_copies).
2. Write a program to create a union employee(id, name, joiningdate(dd, mm, yyyy)). Accept the details for n employees. Accept month and display the details of employee having birthdate in that month.
3. Write a program to create a structure Fraction (numerator, denominator). Accept details of n fractions and write a menu driven program to perform the following operations. Write separate functions for the different options. Use pointer object.
    i. Display the largest fraction
    ii. Display the smallest fraction
    iii. Sort fractions
    iv. Display all

v.

4. Write a program to create a union car (carnumber, carmodelname, carcolor, carcost). Accept details of n cars and write a menu driven program to perform the following operations. Write separate functions for the different options.
    i. Search by name
    ii. Delete by no
    iii. Display all

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program create a structure item(item number, item name, rate, qty, total). Accept details of n items calculate total as qty * rate. And display Bill in the following format ;

Sr. No.	**Item Name**	**Rate**	**Qty**	**Total**
1	Pen	25	3	75.00
2	Eraser	10	1	10.00
			Grand Total :	**85.00**

2. Write a program to create a union movie (name, release year, duration). Accept details of n movies. Write a function to sort them according to release year. Display them in main() function.

3. Write a program to create a structure named company which has name, address, phone and noOfEmployee as member variables. Read name of company, its address, phone and noOfEmployee. Finally display these members‟ value.

4. Write a program to create a structure Mobile (Mobno,MobName, MobCost, MobColor) Accept details of n mobile information and display following;

    i. Display all
    ii. Search for specific item
    iii. Sort according to cost

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 10 : Command Line Arguments

Objective:

To access command-line arguments

Reading:

You should read following topics before starting this exercise

1. Passing arguments from the command line to main
2. Accessing command line arguments
3. Functions - atoi(), atol() and atof()
4. Arithmetic operation on arguments

Ready References:

Command Line Arguments

Command line arguments are a powerful way to pass parameters to a program when it is executed. When the program is executed, it is possible to provide data from the command line to the main() function rather than calling the input statement from within the program. We refer to these values as command line arguments.

Main function with arguments,

```
int main(int argc, char *argv[] )
```

- argc (Argument Count) :

  It is int and stores number of command-line arguments passed by the user including the name of the program.So if we pass a value to a program, value of argc would be 2 (one for argument and one for program name).The value of argc should be non negative.

- argv(Argument Vector) :

  It is array of character pointers listing all the arguments. If argc is greater than zero,the array elements from argv[0] to argv[argc-1] will contain pointers to strings. argv[0] is the name of the program , After that till argv[argc-1] every element is command -line arguments.

atoi() Function

To use the library function atoi() that converts the string representation of a number to an integer.

Syntax :

```
int atoi(const char* string);
```

atol() Function

It converts the string str to a value of type long int by interpreting the characters of the string as numerical values.

Syntax :

```
long int atol ( const char * str );
```

atof() Function

It converts the C-string str to a value of type double by interpreting the characters of the string as numerical values.

Syntax :

```
double atof ( const char * str );
```

/*Ex1. Write a program to display program name and flower name using command line arguments*/

```
#include <stdio.h>
void main(int argc, char *argv[] )
  {
   printf("Zero argument is: %s\n", argv[0]);
   if(argc < 2)
    {
     printf("No argument passed through command line.\n");
    }
   else
    {
     printf("First argument is: %s\n", argv[1]);
    }
}
/*OUTPUT
[root@localhost ADV_C]# cc cmd1.c
[root@localhost ADV_C]# ./a.out India
Zero argument is: ./a.out
First argument is: India
*/
```

/*Ex2. Write a program to display all command line arguments inputs*/

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int i;
    if (argc < 2)
    {
        printf("\n\nNo argument passed through command line!");
    }
     else
     {
        printf("\nArgument are\n");
        for (i = 1; i < argc; i++)
         {
            printf("argv[%d] = %s\n",i,argv[i]);
         }
     }
  return 0;
}
```

```
/*OUTPUT
[root@localhost ADV_C]# cc cmd2.c
[root@localhost ADV_C]# ./a.out monday 100 rose sunday
Argument are
argv[1] = monday
argv[2] = 100
argv[3] = rose
argv[4] = sunday
*/
```

/*Ex3. Write a program to display sum of two integer numbers using command line arguments*/

```
#include <stdio.h>
int main(int argc, char *argv[])
{
        int a,b,sum;
        if(argc!=3)
      {

                printf("not enter the numbers\n");
                return -1;
       }

        a = atoi(argv[1]);
        b = atoi(argv[2]);
        sum = a + b;
        printf("Sum of %d, %d is: %d\n",a,b,sum);
    return 0;

}
/*OUTPUT
[root@localhost ADV_C]# cc cmd3.c
[root@localhost ADV_C]# ./a.out 77 33
Sum of 77, 33 is: 110
*/
```

/*Ex4. Write a program to find the sum of N integer numbers using command line arguments*/

```
#include <stdio.h>
int main(int argc, char *argv[])
{
        int a,b,sum;
        int i; //for loop counter
        if(argc<2)
```

```
        {
                printf("Not inputs\n");
                return -1;
        }
        sum=0;
        for(i=1; i<argc; i++)
        {
                printf("arg[%2d]: %d\n",i,atoi(argv[i]));
                sum  = sum +  atoi(argv[i]);
        }
        printf("SUM of all values: %d\n",sum);
        return 0;
}
/*OUTPUT
[root@localhost ADV_C]# cc cmd4.c
[root@localhost ADV_C]# ./a.out 16 20 7 15 12
arg[ 1]: 16
arg[ 2]: 20
arg[ 3]: 7
arg[ 4]: 15
arg[ 5]: 12
SUM of all values: 70
*/
```

Set A : Write programs to solve the following problems.

1. Write a program to calculate multiplication of two numbers. Pass the numbers as command line arguments.
2. Write a program to accept three integers as command line arguments and find the minimum,maximum and average of the three. Display error message if invalid number of arguments are entered.
3. Write a program to accept ‘n’ integers as command line arguments and display the sum of ‘n’ numbers.
4. Write a program calculate the power of number, which raise its first command line argument to the power of its second command line argument.
5. Write a program to display all command line arguments passed to main in the reverse order.

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program to count the total number of words in the sentence using command line argument.
2. Write a program which accepts a string and characters as command line arguments and replace all occurrences of the character ‘t’ by the symbol “*”.
3. Write a program to find sum of first ‘n’ numbers given in command line arguments recursively (eg ./a.out 1 2 3 4 sum is 10).
4. Write a program to accepts the city names as command line arguments, sort them and display them in ascending order.
5. Write a program to concatenate two strings using command line arguments.

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 11 : File Handling

Objective:
To demonstrate text and random-access files using C.
To implement library, I/O function and operation on files.

Reading:
You should read following topics before starting this exercise

1. Concept of streams
2. Declaring a file pointer
3. Opening and closing a file
4. Reading and Writing to a text file
5. Accessing file using Command line arguments

Ready References:
File
A file is a group of information kept in secondary memory. Up until now, data was input into the programs using the keyboard. Thus, data that may be processed by the programs is stored in files. File operations including creating, opening, writing, reading, renaming, and deleting files is known as file handling in C. We can save and retrieve data in and out of files within our software by using these functions to conduct file operations.

FILE Pointer :
A file pointer to store the reference of the FILE structure returned by the fopen() function. The file pointer is required for all file-handling operations.
Syntax :

```
FILE* file_pointer;
```

Example : FILE *fp;

Operations performed	**Syntax**	**Example**
Declaring File pointer	FILE * pointer;	FILE *fp;
Opening a File	fopen("filename",mode); where mode = "r", "w","a", "r+", "w+", "a+"	fp=fopen("a.txt", "r");
Checking for successful open	if (pointer==NULL)	if(fp==NULL) exit(0);
Checking for end of file	feof	if(feof(fp)) printf("File has ended");
Closing a File	fclose(pointer); fcloseall();	fclose(fp);

Character I/O	fgetc, fscanf
	fputc, fprintf
		ch=fgetc(fp);
		fscanf(fp, ”%c”,&ch);
		fputc(fp,ch);

String I/O	fgets, fscanf fputs, fprintf	fgets(fp,str,80); fscanf(fp, ”%s”,str);
Reading and writing formatted data	fscanf fprintf	fscanf(fp, ”%d%s”,&num,str); fprintf(fp, “%d\t%s\n”, num, str);
Random access tofiles	ftell, fseek, rewind	fseek(fp,0,SEEK_END); /* end of file*/long int size = ftell(fp);

Mode	**Description**
r	opens a text file in read mode
w	opens a text file in write mode
a	opens a text file in append mode
r+	opens a text file in read and write mode
w+	opens a text file in read and write mode
a+	opens a text file in read and write mode

/*Ex1. Write a program to read the number from a text file*/

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int num;
  FILE *fptr;
  if ((fptr = fopen("program.txt","r")) == NULL)
   {
     printf("Error! opening file");
     // Program exits if the file pointer returns NULL.
     exit(1);
   }
    fscanf(fptr,"%d", &num);
    printf("Value of n=%d", num);
    fclose(fptr);
  return 0;
}
/*OUTPUT
[root@localhost ADV_C]# cc file1.c
[root@localhost ADV_C]# ./a.out
Value of n=2808
*/
```

/*Ex2. Write a program to write number into the text file*/

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int num;
  FILE *fptr;
  // use appropriate location
  fptr = fopen("program.txt","w");
  if(fptr == NULL)
  {
    printf("Error!");
    exit(1);
  }
  printf("Enter num: ");
  scanf("%d",&num);
  fprintf(fptr,"%d",num);
  fclose(fptr);
  return 0;
}
/*OUTPUT
[root@localhost ADV_C]# cc file3.c
[root@localhost ADV_C]# ./a.out
Enter num: 99
*/
```

/*Ex3. Write a program to read the file name from user and display the contents of file*/

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  char ch, file_name[25];
  FILE *fp;
  printf("Enter name of a file\n");
  gets(file_name);
  fp = fopen(file_name, "r"); // read mode
  if (fp == NULL)
  {
    perror("Error while opening the file.\n");
    exit(EXIT_FAILURE);
```

```
   }
   printf("The contents of %s file are:\n", file_name);
   while((ch = fgetc(fp)) != EOF)
      printf("%c", ch);
   fclose(fp);
  return 0;
}
/*OUTPUT
[root@localhost ADV_C]# cc file2.c
[root@localhost ADV_C]# ./a.out
Enter name of a file
demo.txt
The contents of demo.txt file are:
Good
Great
Happy
*/
```

/*Ex4. Write a program to write contents into text file.*/

```
#include <stdio.h>
void main()
{
FILE * fp;
char str[20];
int num;
  fp = fopen("book.txt", "w+");
   if(fp==NULL)
   {
    printf("File opening error");
    exit(0);
   }
    fprintf(fp,"%s\t%d\n","ANSI_C",5000);
    fprintf(fp,"%s\t%d\n","LetUSC",7000);
    fprintf(fp,"%s\t%d\n","Python",9000);
    rewind(fp);
     while( !feof(fp))
     {
       fscanf(fp,"%s%d",str,&num);
       printf("%s\t%d\n", str, num);
     }
fclose(fp);
}
/*OUTPUT
[root@localhost ADV_C]# cc file5.c
[root@localhost ADV_C]# ./a.out
```

```
ANSI_C      5000
LetUSC      7000
Python 9000*/
```

```
/*Ex5. Write a program to copy the contents of source file into target file*/

#include <stdio.h>
#include <stdlib.h>
int main()
{
  char ch, source_file[20], target_file[20];
  FILE *source, *target;
  printf("Enter name of file to copy\n");
  gets(source_file);
  source = fopen(source_file, "r");
  if (source == NULL)
  {
    printf("Press any key to exit...\n");
    exit(EXIT_FAILURE);
  }
  printf("Enter name of target file\n");
  gets(target_file);
  target = fopen(target_file, "w");
  if (target == NULL)
  {
    fclose(source);
    printf("Press any key to exit...\n");
    exit(EXIT_FAILURE);
  }
  while ((ch = fgetc(source)) != EOF)
    fputc(ch, target);
  printf("File copied successfully.\n");
  fclose(source);
  fclose(target);
  return 0;
}
/*OUTPUT
[root@localhost ADV_C]# cc file4.c
[root@localhost ADV_C]# ./a.out
Enter name of file to copy
source.txt
Enter name of target file
target.txt
File copied successfully.
*/
```

Set A : Write programs to solve the following problems.

1. Write a program to accept a file name and a single character. Display the count indicating total number of times character occurs in the file.
2. Write a program to accept two filenames as command line arguments. Copy the contents of thefirst file to the second such that the case of all alphabets is reversed.
3. Write a program to accept a filename as command line argument and count the number ofwords, lines and characters in the file.
4. Write a program to read integers from a file and write even and odd numbers in separate files.
5. Write a program to compare two files character by character and check whether they are same or not?

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program to accept details of n employee (emp_number, emp_name, emp_salary) and write it to a file named "employee.txt". Accept employee number from the user and search the employee in the file. Also display the employee details having the highest salary.
2. Write a program which accepts a filename and an integer as command line arguments and encrypts the file using the key. (Use any encryption algorithm).
3. Write a program two read text files and perform the following actions till user selectsexit.
    i. Concatenate two files and save in third file.
    ii. Merge the files.
    iii. Exit
4. Write a menu driven program for a simple text editor to perform the following operations on afile, which contains lines of text.
    i. Display the file
    ii. Copy m lines from position n to p
    iii. Delete m lines from position p
    iv. Modify the nth line
    v. Add n lines

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Assignment 12 : Preprocessor

Objective:

To understand and execute Preprocessor and Format of preprocessor directive

Reading:

You should read following topics before starting this exercise

1. Preprocessor and File inclusion directives
2. Macro substitution directive

Ready References:

Preprocessor

- preprocessor is a micro processor that is used by compiler to transform your code before compilation.
- It is called micro preprocessor because it allows us to add macros.
- preprocessor directives begin with a '#' (hash) symbol.
- The '#' symbol indicates that whatever statement starts with a '#' will go to the preprocessor program to get executed.

Preprocessor Directives

Preprocessor Directives	Description
#define	Used to define a macro
#include	Used to include a file in the source code program
#ifdef	Used to include a section of code if a certain macro is defined by #define
#ifndef	Used to include a section of code if a certain macro is not defined by #define
#if	Check for the specified condition
#else	Alternate code that executes when #if fails
#elif	Combines else and if for another condition check
#endif	Used to mark the end of #if, #ifdef, and #ifndef

File inclusion directives (#include)
#include

- The #include preprocessor directive is used to paste code of given file into current file.
- It is used include system-defined and user-defined header files.
- If included file is not found, compiler renders error.

There are two variants to use #include directive.

- #include <filename>
- #include "filename"

The **#include <filename>** tells the compiler to look for the directory where system header files are held. In UNIX, it is \usr\include directory.

Syntax

```
#include <file_name>
```

/*Ex1. Write a program to display message.*/

```
#include<stdio.h>
 int main()
{
   printf("Education is the most powerful weapon you can use to change the world.");
   return 0;
 }
```

Output :

Education is the most powerful weapon you can use to change the world.

The **#include "filename"** tells the compiler to look in the current directory from where program is running.

Syntax

```
#include "filename"
```

/*Ex2. Write a program to display message.*/

```
#include “stdio.h”
int main()
 {
     printf("The beautiful thing about learning is that no one can take it away from you.");
     return 0;
 }
```

Output :

The beautiful thing about learning is that no one can take it away from you.

/*Ex3. Write a program to display message using #if.*/

```
#include <stdio.h>
#if !defined (MSG)
  #define MSG "GOOD LUCK!!!"
#endif
int main()
{
    printf("WISH YOU %s\n", MSG);
    return 0;
}
```

Output :
WISH YOU GOOD LUCK!!!

Macro Substitution Directive
#define

- Macro substitution is a process where an identifier in a program is replaced by a predefined string composed of one or more tokens.
- The preprocessor accomplishes this task under the direction of **#define** statement.

Syntax
#define first_part second_part
Example

```
#define MAX_SIZE 100
#define PI 3.14159
```

/*Ex4. Write a program to display square of given number.*/

```
#include<stdio.h>
#define square(a) a*a
int main()
{
    int n,sq;
    printf("Enter the number = ");
    scanf("%d",&n);
    sq = square(n);      //replaces sq=n*n before execution of program
    printf("Square of number = %d",sq);
    return 0;
}
```

Output :
Enter the number = 9
Square of number = 81

Macro with Arguments and Nested macro

A macro with arguments in C is a function-like macro, which allows you to define macros that can take input parameters (or arguments) and perform operations similar to a function.

/*Ex5. Write a program to display maximum number out of given two number.*/

```
#include <stdio.h>
#define MAX(a, b) ((a) > (b) ? (a) : (b))      // Defining a macro with arguments

int main()
{
int n1, n2;
printf("\nEnter the numbers = ");
scanf("%d%d",&n1,&n2);
int maximum = MAX(n1, n2);     // Using the macro

printf("The maximum of %d and %d is %d\n", n1, n2, maximum);
return 0;
}
```

Output :

```
Enter the numbers = 44 88
The maximum of 44 and 88 is 88
```

Nested Macro

A nested macro instruction definition is a macro instruction definition you can specify as a set of model statements in the body of an enclosing macro definition.

/*Ex6. Write a program to display area of circle.*/

```
#include <stdio.h>
#define PI 3.1415                    // defining macros
#define circle_Area(r) (PI*r*r)

int main()
{
   float radius, area;
   printf("Enter the radius = ");
   scanf("%f", &radius);
   area = circle_Area(radius);
   printf("Area of Circle = %5.2f", area);
   return 0;
}
```

Output :

```
Enter the radius = 3
Area of Circle = 28.27
```

Set A : Write programs to solve the following problems.

1. Write a program to accept the number and define a macro CUBE for calculating cube of number.
2. Write a program to define value of PI and use it to find perimeter of a circle.
3. Write a program to define a macro one parameter to compute the volume of a sphere.
4. Write a program to display all command line arguments passed to main in the reverse order.
5. Write a program to accept three integers as command line arguments and find the minimum, maximum and average of the three. Display error message if invalid number of arguments are entered.

Signature of the Instructor | | Date | |

Set B : Write programs to solve the following problems.

1. Write a program which accepts a string and display upper and lower case of given string.
2. Write a program to implement a macro named NOTEQUAL that performs logical not operation.
3. Write a program which accepts a string and two characters as command line arguments andreplace all occurrences of the first character by the second.
4. Write a program which accepts a two integers as command line arguments. If second number is 1 then display factorial of first number and if second number is 2 the check whether first number is even or odd. If the user enters invalid number of arguments, display appropriate message.
5. Create a header file "mymacros.h" which defines the following macros.
    a) SQR(x)
    b) GREATER2(x,y)
    c) GREATER3 (x,y,z) –nested
    d) FLAG (value=1) (which may or may not be defined)

Include this file in your program. Write a menu driven program to use macros SQR, GREATER2 and GREATER3. Your program should run the first two macros if the macro called FLAG has been defined. If it is not defined, execute the other two macros. Run the program twice – with FLAG defined and with FLAG not defined.

Signature of the Instructor | | Date | |

Assignment Evaluation

0: Not Done		1: Incomplete		2:Late Complete	
3: Needs Improvement		4: Complete		5: Well Done	

Practical In-Charge

Case Study or Mini Project

a)

The Cryptarithmetic puzzle

Cryptarithmetic" puzzles are puzzles in which one gets problems like these

```
  hello
+ there
---------
```

world

and is asked to assign digits to each letter so that the resulting addition is correct. Each digit from 0to 9 must be used at most once, and the leading digits may not be 0. In the above cases, for example, we can get the solutions

```
  56442
+ 15606
-------
  72048
```

Write a program to find a solution to cryptarithmetic problems for which the input consists of triples of strings each containing up to 128 lower-case letters and the output is in the form given inthe sample below.

For the input Produce the output

hello there world

```
  hello     56442
+ there   + 15606
-----------------
  world     72048
```

b)

The library problem

Write a program to solve the following problem: A library wants a program that will calculate the duedate for a book on the basis of the issue date. The no. of days for which the book is issued is decided by the librarian at the time of issuing the book.

For e.g. If the librarian enters the current date as 14-02-2024 and the no of days in which the book is due as 15, then your program should calculate the due date and give the output as 29-02-2024. Your program should accept the current date (day, month, year) and the number of issue days as input and generate the due date as output.

c)

The Secret Word problem

You have determined that the enemy is using the following mechanism to encode secret words. You believe that the first letters of each word in enemy messages form secret words.

Only the first letters of consecutive words are used to form the secret words. Further, a sentence may contain other words before and/or after the actual words that make up the secret word. Messages always contain a single space between words.

Write a program that takes a secret word and a message as input and determines if the message contains the secret word. The program should not be case-sensitive and should ignore punctuation.

Input Format : The first line of input consists of the secret word.
The second line contains thesentence to check.

Output Format:
The output will be “Secret word found” if the secret word is found in the sentence,otherwise, output “Secret word not found.”

Input:
year
the yellow elephant ate raw bananas
Output:
Secret word found

Input:
you
you will often fail unless you try harder
Output:
Secret word not found

d)

Hotel Management System

This program is adapted to provide us information on reserving rooms, book an event, check the features etc. This program is untroublesome as it will serve the admin or user to be updated about the records without any strain and it is favored much by the people involved in the business sector. As we are aware of the busy and hectic schedule of business people, this Hotel management system turns out to be a great relief for them. This program definitely has a wide scope to minimize errors in the making of bills and it also limits the delay of delivering bills to the customers which can include taxes on the basis of their expenditure.

In the making of this program, the users or admins comfort and reliability is put into consideration which focuses to save one's quality time. This system has very least chance of data loss and we don't have to worry about it being damaged

e)

The Credit Card Verification Problem

You are provided with a credit card number with the length varying from 13 digits to 16 digits. Each digit of the credit card is weighted by either 2 or 1. The credit card number must be zero-filled on the left to create a sixteen digit number, and then the pattern starts with 2, alternating with a 1. If thenumber multiplied by the weight results in a 2-digit number, each digit is added to the sum. The finalsum with the check digit should be a multiple of 10.

Example:

5	4	9	9	0	0	1	1	0	0	1	2	0	0	3	4	
2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	
1+0	+4	+1+8	+9	+0	+0	+2	+1	+0	+0	+2	+2	+0	+0	+6	+4	=40

40 mod 10 = 0. If the last digit did not result in a number divisible by 10, the credit card number is invalid.

Input Format :

The user will provide you with a credit card number without spaces.

Output Format :

The program will return "Valid" or "Invalid" depending on the success or failure ofthe check digit.

Input:
5499001100120034
Output:
Valid

Input:
5499001100120036
Output:
Invalid