

CPTC 2017 (ODD)**GROUP – “A”**

1	a	6	c	11	b	16	d
2	d	7	b	12	c	17	c
3	a	8	a	13	d	18	d
4	c	9	b	14	a	19	a
5	b	10	c	15	d	20	c

GROUP – “B”

Q.2. Define flowchart, flowchart symbols and its importance.

(4)

Ans. : Flowchart is the pictorial representation of the solution of any problem which is used to represent the specified logics of algorithm using ANSI standard flowcharting symbols.

Like algorithms, Flowchart is also a most basic program design and system development tool that is used to represent the algorithmic steps of solutions of a problem using diagrams.

Flowcharts help programmers to understand the flow of logic and control transferring concept in the program, making easy the program development.

ANSI Standard Flowchart Symbols :

Terminals : Every Algorithm must has a START and a STOP step. START is always first step whereas STOP is last step. These are called Terminal Symbols.



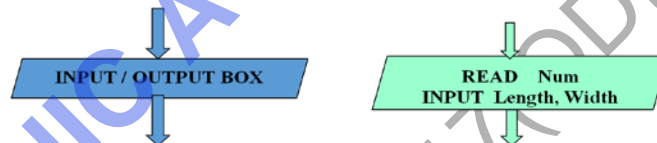
Variable Declaration : Every Algorithmic solution needs to store and use data. Data are stored in variables. Variable needs to be introduced about its name and type of data to be stored in it. DECLARE word is used to do this task followed by Variable Name and Data Type.



Assignment / Process Box : Every Algorithmic solution needs to consider some data to be stored to use to perform task. This action is called Assignment. Assignment is performed by putting a value in a named variable. LET or ASSIGN is used to perform this task.



INPUT : Solution of your problem needs data to be used to process the task. Receiving data from external input source like Keyboard, Disk, or Mouse to specified memory variable is known as INPUT. Input is performed using READ statement.

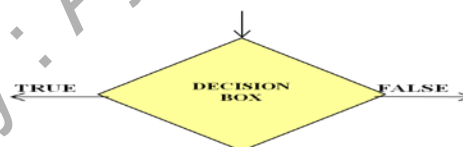


OUTPUT : After completing the task one or more results should be shown. Results are stored in Output Variables. Displaying the result on Output Device like Screen or Paper is known as OUTPUT. Output operation is shown by WRITE or PRINT statement.

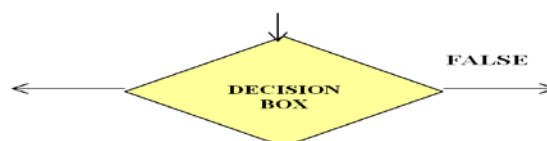


Decision Making Statements

In some cases it is needed to take decision among two or more options. At that situation Decision Making construct is used. An expression is checked logically to get one of the True or False output, depending upon which control transfer operation performed. To perform Decision Making operation IF..., IF....ELSE.... or IF....ELSEIF.... construct is used.



Iteration (Looping) : Some of the solutions need to perform tasks to repeat up to a finite number of times, called Iteration . Iteration is the process wherein a computer performs a process over and over again repeatedly for a specific number of times or until a specific condition has met. Iterations are performed using some specific statements called Looping statements. FOR , WHILE and DO are used to perform looping operations.



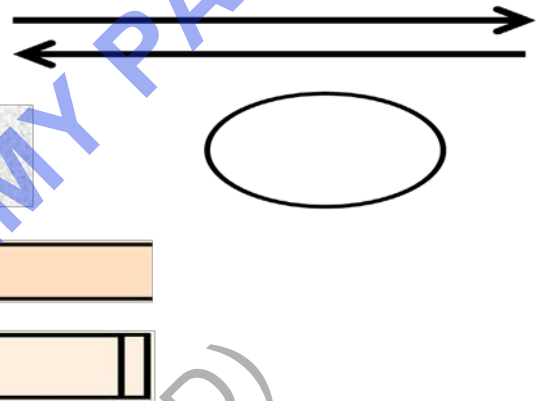
OTHER FLOWCHART SYMBOLS

Lines or arrows represent the direction of the flow of control.

Connector (connect one part of the flowchart to another)

Comments, Explanations, Definitions.

Refers to separate flowchart



Q.2. Write a C program to check odd or even. (4)

Ans. :

// program to check Odd or Even of Number

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int num;
```

```
    printf("\n\n Enter any Integer No. : ");
```

```
    scanf("%d",&num);
```

```
    if(num%2 == 0)
```

```
        printf("\n\n %d is Even Number",num);
```

```
    else
```

```
        printf("\n\n %d is Odd Number",num);
```

```
    return 0;
```

```
}
```

```
Enter any Integer No. : 23
```

```
23 is Odd Number
```

```
Enter any Integer No. : 44
```

```
44 is Even Number
```

Q.3. Write a C program to swap values of two variables without using temporary variable. (4)

Ans.:

// Program to swap values of two variables without using temporary variable

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int a, b;
```

```

printf("\n\n Enter value for A : ");
scanf("%d",&a);
printf("\n\n Enter value for B : ");
scanf("%d",&b);
printf("\n\n Before Swapping A = %d \t B = %d",a,b);
//Exchanging values of two variables
a = a + b;
b = a - b;
a = a - b;
printf("\n\n After Swapping A = %d \t B = %d",a,b);
return 0;
}

```

```

Enter value for A : 10
Enter value for B : 20
Before Swapping A = 10      B = 20
After Swapping A = 20      B = 10

```

Q.3. Write syntax of if.... else statement with program.

(4)

Ans.: The if ()...else... statement is a bi-directional conditional control transfer construct, used to take decision on the given condition when one of the two possible options is to choose. The condition may be either relational or logical expression. If the condition will be TRUE, it will return the execution of statement-block-1; otherwise execute statement-block-2.

Syntax:

```

if ( condition )
    Statement-block-1;
else
    Statement-block-2;

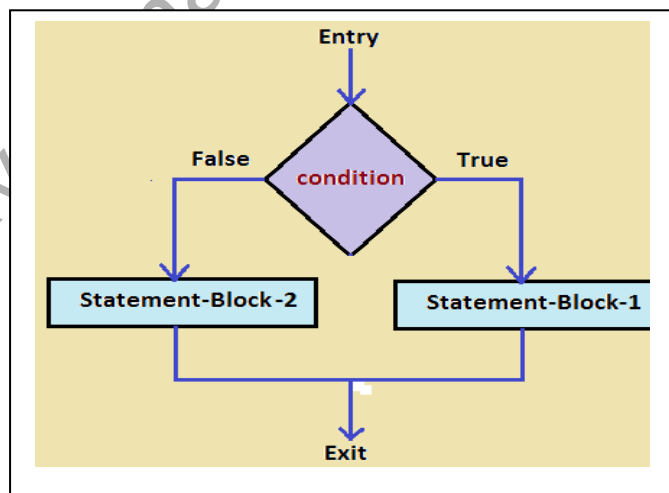
```

Ex:

```

if ( a > b )
    printf("\n%d is bigger",a);
else
    printf("\n%d is bigger",b);

```



Q.4. Define array and show how to declare 2D array in C language.

(4)

Ans. : An **ARRAY** is a collection of homogeneous data elements (i.e., of same data type) described by a single variable name. Each individual element of an array is referred by a **Subscripted Variable** or **Integer Constant**. These subscripted variables or indexed variables are enclosed within square brackets and affixed with array variables.

If single subscript is required to refer to an element of an array, the array is known as One Dimensional or Linear Array. If two subscripts are required to refer to an element of an array, the array is known as Two Dimensional Array and so on.

Note: The no. of dimensions supported in an array depends upon RAM capacity and operating system used in the computer system.

Declaring an ARRAY :→

Syntax :

DataType ArrayName [dim-1][dim-2] ... [dim-n] ;

An array in a program, must contain following information :→

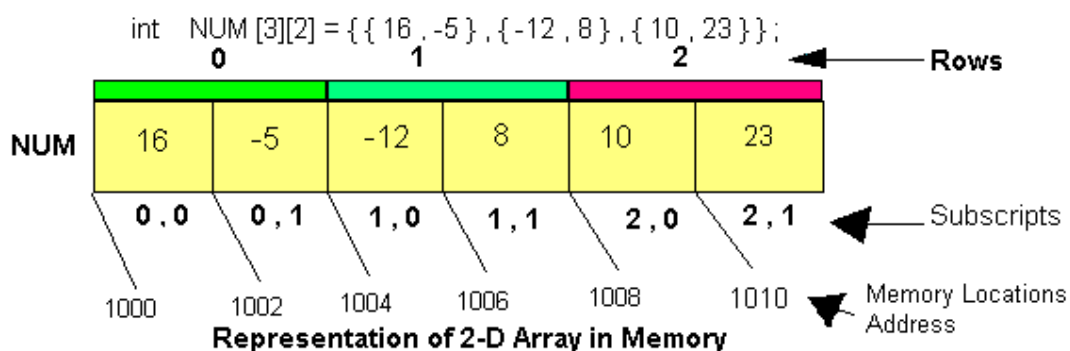
- ♦ The type and name of the array.
- ♦ The number of subscripts in the array.
- ♦ The total number of memory locations to be reserved, or more specifically, the maximum value of each subscript.

A Two Dimensional Array can be treated as array of two parts **ROW** and **COLUMN**. The first subscript represents the ROW position, whereas the second subscript represents the COLUMN position. Similarly, a Three Dimensional Array can be treated as array of two dimensional arrays, in which last two dimensions are treated as one dimensional array.

Declaration of 2D Array :

Datatype array_name[size1][size2];

Ex:



Formula to Calculate the Address of an Element in 2-D Array

= Base Address + Row No. x No. of Columns in a row x Size of (Data Type)

+ Column No. x Size of (Data Type) // Data Type Size is implicitly assumed in the formula

Ex. : $\&\text{num}[1][1] = 1000 + 1 \times 2 \times (2) + 1 \times (2) = 1006$

Q.4. Write down the difference between pointer and array.

(4)

Ans. : Differences between Pointer and Array are:

- An array is a collection of elements of similar data type whereas the pointer is a variable that stores the address of another variable.
- An array size decides the number of variables it can store whereas; a pointer variable can store the address of only one variable in it.
- Arrays can be initialized at the definition, while pointers cannot be initialized at the definition.
- Arrays are static in nature which means once the size of the array is declared, it cannot be resized according to users requirement. Whereas pointers are dynamic in nature, which means the memory allocated can be resized later at any point in time.
- Arrays are allocated at compile time while pointers are allocated at runtime.

Q.5. Write a program in C to check Armstrong number.

(4)

Ans. :

// Program to check armstrong no.

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int n, m, d, r=0;
```

```
    printf("\n\n Enter any 3 digit integer no to check for armstrong no. : ");
```

```
    scanf("%d",&n);
```

```
    m = n;
```

```
    while(m>0)
```

```
    {
```

```
        d = m %10;
```

```
        r = r + (d * d * d);
```

```
        m = m / 10;
```

```
    }
```

```
    if (n == r)
```

```
        printf("\n\n %d is an Armstrong No.",n);
```

```
    else
```

```
        printf("\n\n %d is not an Armstrong No.",n);
```

```
Enter any 3 digit integer no to check for armstrong no. : 371
371 is an Armstrong No.
```

```
Enter any 3 digit integer no to check for armstrong no. : 452
452 is not an Armstrong No.
```

```

    return 0;
}

```

Q.5. Write a C program to find largest of group of any number using array.

(4)

Ans. :

// Program to check largest Number from a list of data in 1-D Array

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
main()
```

```
{
```

```
    int num[100],n,i, big;
```

```
    printf("\n How many data ? ");
```

```
    scanf("%d",&n);
```

```
    printf("\n Enter the %d numbers : \n",n);
```

```
    for(i=0 ; i<n ; i++)
```

```
        scanf("%d",&num[i]);
```

```
    printf("\nOriginal Array is : \n");
```

```
    for(i=0 ; i<n ; i++)
```

```
        printf("\n %5d",num[i]);
```

// Finding the Largest Number

```
    big = num[0];
```

```
    for(i=1;i<n;i++)
```

```
        if (num[i] > big)
```

```
            big = num[i];
```

```
    printf("\n The Largest Number in the List is : %d",big);
```

```
    return 0;
```

```
}
```

```

How many data ? 10
Enter the 10 numbers :
45
23
67
54
12
33
77
49
41
39

Original Array is :
    45
    23
    67
    54
    12
    33
    77
    49
    41
    39
The Largest Number in the List is : 77

```

Q.6. Explain arithmetic and relational operators in C.

(4)

Ans. : ARITHMETIC OPERATORS

Arithmetic Operators are binary nature of operators. These operators need two operands to perform action. They perform mathematical calculations on numeric data and produces output depending upon the types of operands used and the nature of operation performed. 'C' does not support power (exponentiation) operator.

ARITHMETIC OPERATORS				
SYMBOL	OPERATION	OPERANDS	PRECEDENCE	EXAMPLE
%	Remainder	int, long int	Highest	45%7=3
/	Division	int, long int, float, double	Highest	12/5=2.4
*	Multiplication	int, long int, float, double	Highest	3.5*3=10.5
+	Addition	int, long int, float, double	Lowest	13+6=19
-	Subtraction	int, long int, float, double	Lowest	34-9=25

RELATIONAL OPERATORS

Relational Operators are Binary operators that appear between two operands. There is no relative precedence or association among the relational operators. Relational operators with operands form relational expression to compare the values of two operands, returns Logical (Boolean) value either True (1) or False(0).

Relational expressions are used to test a condition in order to decide whether an action should be taken or not. Operands of a relational expression may be any numeric values. In relational expression, an arithmetic value can not be compared with a string value.

Syntax : opnd1 Rel_optr opnd2

RELATIONAL OPERATORS			
OPERATOR	OPERATION PERFORMED	PRECEDENCE	EXAMPLE
<	Less Than	Highest	a < b
<=	Less Than orEqual To	Highest	a <= b
>	Greater Than	Highest	x > y
>=	Greater Thanor Equal To	Highest	m >= n
==	Equal To (Equality)	Lowest	p == q
!=	Not Equal To(InEquality)	Lowest	d != 0

Example:

```
if ( a< b)
    small = a;
else
    small = b;
```


Q.6. What is an operator? Explain logical and assignment operator in C. (4)

Ans. : Operators are the verbs of any programming language. Operators do the action to perform the user computed values in the program. In an expression the operands are subjects and objects of those verbs which are in the form of operators upon which action is to be performed during processing.

Operators are symbols that is used in program to manipulate operands in the form of data and variables embedded in the arithmetic and/or logical expressions.

LOGICAL OPERATORS

A logical operator is used to form a complex test condition in order to take a decision. An expression is formed by logical operators which combine two more relational expressions is termed as Logical Expression or Combined Relational Expression. A logical expression produces logical value Boolean True (1) or False (0). Logical operators produce (return) value according to the logical Truth Table.

LOGICAL OPERATORS			
OPERATOR	OPERATION PERFORMED	PRECEDENCE	EXAMPLE
!	Negation or Logical NOT	Highest	! digit
&&	Conjunction Or Logical AND	Intermediate	(gender=='F') && (age>18)
	Disjunction Or Logical OR	Lowest	(status=='1') (age>=30)

ASSIGNMENT OPERATORS

Assignment Operators are used to assign the result of an expression to a variable usually using basic assignment '=' operator. 'C' provides a set of shorthand assignment operators of the form var opt= exp. This form of shorthand composed of a basic operator in combination with '=' operator.

In an Assignment statement or expression, target variable will be always in left side to assignment operator, whereas the expression will be always on the right side to the assignment operator. The associativity of these operators are from Right to Left, means the expression right to the '=' operator is processed first and then the result is stored in the variable left side of '=' operator.

ASSIGNMENT OPERATORS		
OPERATOR	OPERATION PERFORMED	EXAMPLE
=	Assignment	m = a + x / y
*=	Multiply and Assign	N *= 2 (n = n * 2)
/=	Divide and Assign	Num /= 3 (num = num / 3)

%=	Remainder and Assign	Val %= x (val = val % x)
+=	Add and Assign	N += j (n = n + j)
-=	Subtract and Assign	Num -= 5 (num = num – 5)
&=	Bitwise AND and Assign	A &= b (a = a & b)
^=	Bitwise XOR and Assign	A ^= b (a = a ^ b)
=	Bitwise OR and Assign	A = b (a = a b)
<<=	Bitwise Left Shift and Assign	A <<= 2 (a = a << 2)
>>=	Bitwise Right Shift and Assign	X >>= 1 (x = x >> 1)

GROUP - “C”

Q.7. Write a C program to read a matrix and print its transpose using array.

(6)

Ans.

// Program to Transpose a 2D Matrix

```
#include<stdio.h>
```

```
#define MAX 100
```

```
int main()
```

```
{
```

```
    int a[MAX][MAX], t[MAX][MAX];
```

```
    int i,j,r,c;
```

```
    printf("\n Enter Dimension of Matrix : ");
```

```
    scanf("%d %d",&r,&c);
```

```
    //Input in Matrix
```

```
    printf("\n Enter %d numbers in Source Matrix :\n",r*c);
```

```
    for(i=0;i<r;i++)
```

```
        for(j=0;j<c;j++)
```

```
            scanf("%d",&a[i][j]);
```

```
    //Transposing the matrix
```

```
    for(i=0;i<r;i++)
```

```
        for(j=0;j<c;j++)
```

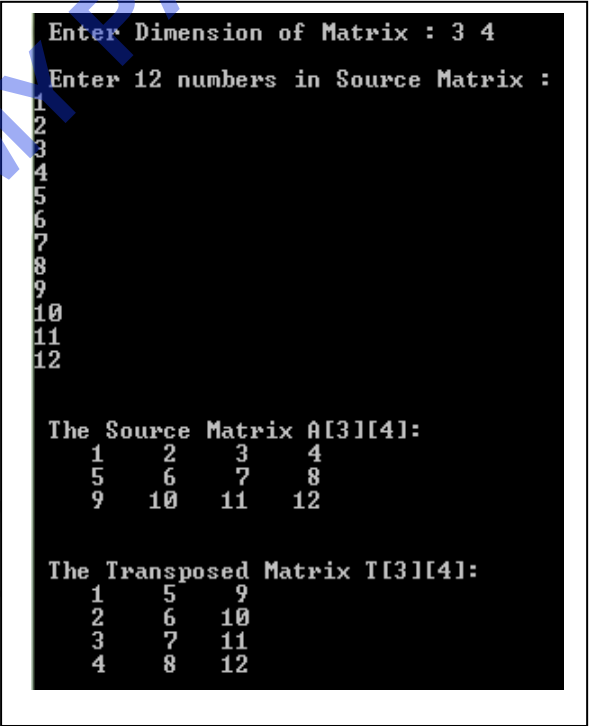
```
            t[j][i]=a[i][j];
```

//Displaying Output

```

printf("\n\n The Source Matrix A[%d][%d]:\n",r,c);
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
    printf("%5d",a[i][j]);
    printf("\n");
}
printf("\n\n The Transposed Matrix T[%d][%d]:\n",r,c);
for(i=0;i<c;i++)
{
    for(j=0;j<r;j++)
    printf("%5d",t[i][j]);
    printf("\n");
}
return 0;
}

```



Enter Dimension of Matrix : 3 4
Enter 12 numbers in Source Matrix :
1
2
3
4
5
6
7
8
9
10
11
12

The Source Matrix A[3][4]:
1 2 3 4
5 6 7 8
9 10 11 12

The Transposed Matrix T[3][4]:
1 5 9
2 6 10
3 7 11
4 8 12

Q.7. Write a C program for addition of two matrix.

(6)

Ans. :

//Addition of 2 Arrays

```
#include<stdio.h>
```

```
#define MAX 100
```

```
int main()
```

```
{
```

```
    int a[MAX][MAX], b[MAX][MAX];
```

```
    int s[MAX][MAX], i,j,r,c;
```

```
    printf("\nEnter Dimension of Matrix : ");
```

```
    scanf("%d %d",&r,&c);
```

//Input in Matrix

```
    printf("\nEnter %d numbers in 1st Matrix : ",r*c);
```

```

for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        scanf("%d",&a[i][j]);
printf("\nEnter %d numbers in 2nd Matrix : ",r*c);
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        scanf("%d",&b[i][j]);
//Adding two matrices
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        s[i][j]=a[i][j] + b[i][j];
//Displaying Output
printf("\n\n1st Matrix A[%d][%d]:\n",r,c);
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
        printf("%5d",a[i][j]);
    printf("\n");
}
printf("\n\n2nd Matrix B[%d][%d]:\n",r,c);
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
        printf("%5d",b[i][j]);
    printf("\n");
}
printf("\n\nResultant Matrix S[%d][%d]:\n",r,c);
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)

```

```

Enter Dimension of Matrix : 3 2
Enter 6 numbers in 1st Matrix : 10 20 30 40 50 60
Enter 6 numbers in 2nd Matrix : 11 22 33 44 55 66

1st Matrix A[3][2]:
10  20
30  40
50  60

2nd Matrix B[3][2]:
11  22
33  44
55  66

Resultant Matrix S[3][2]:
21  42
63  84
105 126

```

```

        printf("%5d",s[i][j]);
        printf("\n");
    }
    return 0;
}

```

Q.8. Write a program to test the given number is prime or not.

(6)

Ans.

//Program to check whether an Integer Number is Prime or Not.

```

#include<stdio.h>
#include<math.h>
main()
{
    int num,j,val,flag=1;
    printf("\n\nEnter any Integer No. : ");
    scanf("%d",&num);
    j=2;
    val=sqrt(num);
    while(j<=val)
    {
        if (num%j==0)
        {
            flag=0;
            break;
        }
        j++;
    }
    if(flag==1)
        printf("\n\n%d is a Prime No.",num);
}

```

```

Enter any Integer No. : 71
71 is a Prime No.

```

```

Enter any Integer No. : 49
49 is not a Prime No.

```

```

else
    printf("\n\n%d is not a Prime No.",num);
}

```

Q.8. Write a C program to find GCD of two numbers using recursion.

(6)

Ans. :

//Program to find GCD of Two Numbers using Recursion

```
#include <stdio.h>
```

```
int gcd(int n1, int n2);
```

```
int main()
```

```
{
```

```
    int n1, n2;
```

```
    printf("\n\nEnter two positive integers: ");
```

```
    scanf("%d %d", &n1, &n2);
```

```
    printf("\nG.C.D of %d and %d is %d.", n1, n2, gcd(n1, n2));
```

```
    return 0;
```

```
}
```

//Recursive Function Definition

```
int gcd(int n1, int n2) {
```

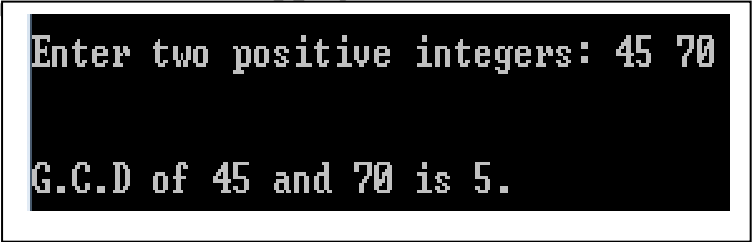
```
    if (n2 != 0)
```

```
        return gcd(n2, n1 % n2);
```

```
    else
```

```
        return n1;
```

```
}
```



```

Enter two positive integers: 45 70

G.C.D of 45 and 70 is 5.

```

Q.9. Explain call by value and call by reference with suitable example.

(6)

Ans. : The mechanism used to pass data to a function is via argument list, where individual arguments are called Actual Arguments. These arguments are enclosed in parentheses after the function name. The actual arguments must correspond in number, type and order with formal arguments specified in the function definition. The actual argument can be constants, variables, array names or expressions.

There are two approaches to passing arguments to a function. These are :--

➤ **Call by Value.**

➤ **Call by Reference.****Call by Value**

In this approach, the names of actual arguments are used in the function call. In this way the values of the actual arguments are passed to the function. When control is transferred to the called function, the values of the actual arguments are substituted to the corresponding formal arguments and the body of the function is executed. If the called function is supposed to return a value, it is returned via return statement.

For Example : `int sum(int a, int b)`

```
{
    int c;
    c = a + b;
    return c;
}
```

Function Call : `int s = sum(x, y);` // Function Call by Value

**Call by Reference**

In this approach, the names of actual arguments are used in the function call. In this way, the addresses of the actual arguments are passed to the function. When control is transferred to the called function, the addresses of the actual arguments are substituted to the corresponding formal arguments and the body of the function is executed. The formal arguments are declared as pointers to types that match the data types of the actual arguments.

This approach is of practical importance while passing arrays and structures among function, and also for passing back more than one value to the calling function.

Example : `void swap(int *a, int *b)`

```
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
    return;
}
```

Function Call : `int x=56, y=34;`
`swap(&x, &y);` // Function Call by Reference

Q.9. Write a program in C to show the following pattern.

(6)

```

      A
     ABC
    ABCDE
   ABCDEFG
  ABCDEFGHI

```

Ans. :

// Program to print Alphabetic Pyramid Shape

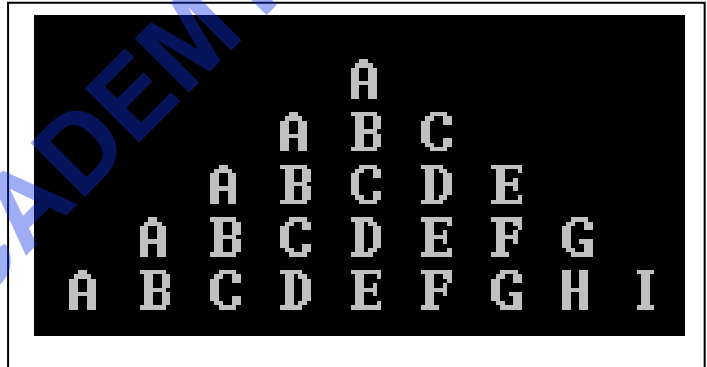
```
#include<stdio.h>
```

```
int main()
```

```

{
    int r, c, b, n=65;
    for(r = 1 ; r <= 5 ; r++)
    {
        printf("\n");
        for(b=4 ; b >= r ; b--)
            printf("%2c", " ");
        for(c = 65 ; c <= n ; c++)
            printf("%2c", (char)c);
        n = n + 2;
    }
    return 0;
}

```



Q.10. Explain switch statement with syntax and example. (6)

Ans. : The **switch** statement causes a particular group of statement to be chosen from several available groups expressed under **case** followed by a **value**. The selection is based upon the current value of an expression that is included within the switch statement and control is transferred in that section of the matching **case value**.

Syntax :

```

switch ( expression )
{
    case val-1 :
        statement block-1;
        break;
    case val-2 :
        statement block-2;
        break;
    :
    :
    case val-N :
        statement block-N;
        break;
    default :
        statement-block-default
}

```

Where expression takes any given value from val-1, val-2, , val-N, the control is transferred to that appropriate case.

In each case, the statements are executed and then the break statement transfers the control out of switch statement.

The default keyword is, usually mentioned at the end of switch statement, used if the value of the expression does not match any of the case values.

//Example Program Name : WeekDay.c

//Program to display day name of given weekday no.

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int dayNo;
```

```
    printf("\n\nEnter any day no. of a Week [0..6] : ");
```

```
    scanf("%d",&dayNo);
```

```
    switch(dayNo)
```

```
    {
```

```
        case 0: printf("\n\nThe day of week is : Sunday");
```

```
            break;
```

```
        case 1: printf("\n\nThe day of week is : Monday");
```

```
            break;
```

```
        case 2: printf("\n\nThe day of week is : Tuesday");
```

```
            break;
```

```
        case 3: printf("\n\nThe day of week is : Wednesday");
```

```
            break;
```

```
        case 4: printf("\n\nThe day of week is : Thursday");
```

```
            break;
```

```
        case 5: printf("\n\nThe day of week is : Friday");
```

```
            break;
```

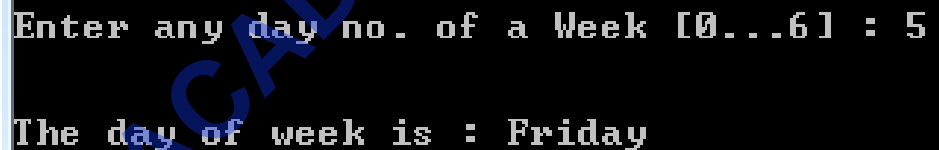
```
        case 6: printf("\n\nThe day of week is : Saturday");
```

```
            break;
```

```
        default : printf("\n\nSorry ! Wrong Input...");
```

```
    }
```

```
}
```



```
Enter any day no. of a Week [0...6] : 5

The day of week is : Friday
```

Q.10. Write a C program to find the area and perimeter of a circle using switch statement. (6)

Ans. :

// Program to calculate area and perimeter of a circle

```
#include<stdio.h>
```

```
# define PI 22/7
```

```
int main()
```

```
{
```

```
    float rad, area, perimeter;
```

```
    printf("\n\n Enter size of Radius of circle : ");
```

```
    scanf("%f",&rad);
```

```
    area = PI * rad * rad;
```

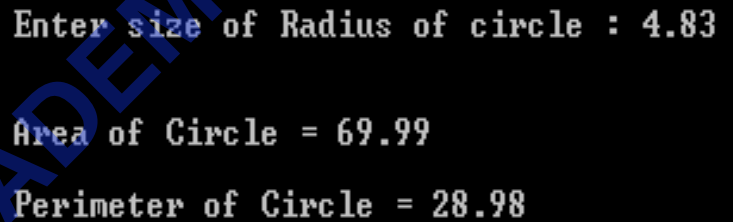
```
    perimeter = 2 * PI * rad;
```

```
    printf("\n\n Area of Circle = %.2f",area);
```

```
    printf("\n\n Perimeter of Circle = %.2f",perimeter);
```

```
    return 0;
```

```
}
```



```
Enter size of Radius of circle : 4.83
Area of Circle = 69.99
Perimeter of Circle = 28.98
```

Q.11. Write a C program to calculate factorial and also check even or odd number using function. (6)

Ans. :

// Program to Calculate factorial, and Odd or Even of a Number

```
#include<stdio.h>
```

```
main()
```

```
{
```

```
    int odd_even(int);
```

```
    long int factorial(int);
```

```
    int n, num, flag;
```

```
    long int z;
```

```
    printf("\n\n Enter any Integer No. to calculate factorial [ < 13 ]: ");
```

```
    scanf("%d",&n);
```

```
    if(n<=1)
```

```
        printf("\n Factorial of %d = 1",n);
```

```

else
{
    z = factorial(n);
    printf("\n Factorial of %d = %ld",n,z);
}
printf("\n\n Enter an Integer No. to Check Odd or Even : ");
scanf("%d",&num);
flag = odd_even(num);
if(flag == 1)
    printf("\n %d is an Odd Number",num);
else
    printf("\n %d is an Even Number",num);
return 0;
}

```

//Function to Check Odd or Even

```
int odd_even(int m)
```

```

{
    if(m%2 == 0)
        return 0;
    else
        return 1;
}

```

//Function to calculate Factorial

```
long int factorial(int m)
```

```

{
    int j;
    long int p=1;
    for(j=1 ; j<=m ; j++)
        p *= j;
    return p;
}

```

```

Enter any Integer No. to calculate factorial [ < 13 ]: 7
Factorial of 7 = 5040
Enter an Integer No. to Check Odd or Even : 56
56 is an Even Number

```

}

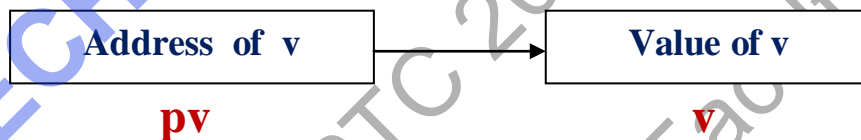
Q.11. Explain the declaration syntax of pointer, and pointer initialization with example. (6)

Ans. : A Pointer is a variable that represents the location (rather than the value) of a data item within the memory, such as a variable or an array element. Pointers can be used to pass information back and forth between a function and its reference point. Pointers provide a way to return multiple data items from a function via function arguments. Pointers also permit references to other functions to be specified as arguments to a given function. This has the effect of passing functions as arguments to the given function.

Every data items stored within memory. It occupies one or more contiguous memory cells (i.e., adjacent words or bytes). The number of memory cells required for storing data item depends on the type of data items. The data item can be accessed by its address within memory. This address of location can be determined by a variable preceded by '&' called address operator, that evaluates the address of its operand.

Let us assign the address of a variable say v to another variable pv.

pv = &v will store the address of variable v into pv.



Where & is a **Unary Operator** always printed by '%u' format string.

Pointer variables must be declared before use. When a pointer variable is declared, the variable name must be preceded by an **Asterisk (*)**. This identifies the fact that the variable is a pointer. The data type that appears in the declaration refers to the Object of the pointer, i.e., the item that is stored in the address represented by the pointer, rather than the pointer itself.

Syntax :

data-type *pointer-variable-name ;

Ex.--

int *ptr;

int x, *px;

px = &x; // Storing address of x into pointer variable px

Pointer Initialization

Datatype *pointer_name = value;

Ex: int *p = 45;

printf("\nValue of pointer P = %d is stored at address %u", *p, p);
