



**Important Instructions to examiners:**

- 1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
- 2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
- 3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
- 4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
- 5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
- 6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
- 7) For programming language papers, credit may be given to any other program based on equivalent concept.

**Q.1] a) (01-mark for declaration, 01-mark for initialization)**

i. Variable declaration tells the compiler what the variable name is and what type of data the variable will hold.

e.g. int a;

ii. During variable initialization a variable is assigned some value.

e.g. int a=10;

**Q.1] b) (01-mark for syntax, 01-mark for example)**

A conditional operator is used to construct expressions of the form

**exp1? exp2: exp3**

where exp1, exp2 and exp3 are expressions.

e.g. int a=10,b=5,x;

x=(a>b) ? a : b;

**Q.1] c) Relational operators with their meaning (Any 4 :- 1/2 mark each):**

Relational Operators	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

**Q.1] d) (02-marks for definition)**

The continue statement is used to transfer the control at the beginning of loop. Wherever the continue statement is encountered immediately the control passes at the beginning of the loop. Continue is generally used with if statement.

**Q.1] e) (02-marks for definition)**

The break statement breaks the the loop and control is transferred outside the loop. Break doesn't allow to continue the loop in which it is written.



***Q.1] f) Give syntax of nested if else statement (02-marks for syntax)***

if(test condition1)

```
{
    if(test condition2)
    {
        statement-1;
    }
    else
    {
        statement-2;
    }
}
else
{
    statement-3;
}
statement-x;
```

***Q.1] g) State the use and syntax of strlen() function (01-mark for definition, 01-mark for example)***

strlen() function counts and returns the number of characters in a string.

syntax: n=strlen(string);

where n is an integer variable, which receives the value of the length of the string.

***Q.1] h) Define array (02-marks for definition)***

An array is fixed-size sequenced collection of elements of the same data type. It is simply a grouping of same type data.

***Q.1] i) Define function prototype. (01-mark for definition, 01-mark for format)***

Function prototype is a function declaration before it is invoked. It consists of function type, function name, parameter list and terminating semicolon. It is coded in the following format:

```
function_type function_name(parameter list);
```



***Q.1] j) Define function (02-marks for types)***

A function is a self contained block of code that performs a particular task.

There are two types of functions:

- i. Library functions(e.g. printf(),scanf())
- ii. User defined functions(e.g. add())

***Q.1] k) Define call by value. (02-marks for definition)***

When a function is called by passing actual parameters in function name then it is called as call by value.

***Q.1] l) What do you mean by structure? Give syntax of declaring it (01-mark for definition, 01-mark for syntax)***

Structure is a collection of logically related data items of same or different data types.

**Syntax to declare a structure:**

```
struct struct_name
{
    data_type member1;
    data_type member2;
    -----
    -----
    -----
    data_type member n;
};
```

***Q.1] m) what is pointer? Give its declaration. (01-mark for definition, 01-mark for declaration)***

Pointer is a variable which stores the address of another variable.

Declaration of pointer:

```
data_type *ptr_name;
```

***Q.1] n) Define pointer variable. What are the advantages of pointer? (01-mark for definition, ½ mark each for any two advantages)***

Pointer is a variable which stores the address of another variable.

Advantages of pointer:

- i. It saves memory space.
- ii. It processes data very fast.
- iii. Programs with pointers are very efficient.



- iv. It is used to help locating exact value at exact locations
- v. It helps in passing array to a function in a simple way.
- vi. Function handling becomes efficient with pointers.

***Q.2] a) State four rules for choosing variable name. (01-mark each, any four rules)***

***Four rules for choosing variable name:***

- i. It must begin with a letter. Some systems permit underscore as the first character.
- ii. ANSI standard recognizes a length of 31 characters. However length of variable should not normally more than eight characters, since only the first characters are treated as significant by many compilers.
- iii. Uppercase and lowercase are significant. That is, the variable Total is not the same as total or TOTAL.
- iv. Variable should not be a keyword.
- v. In variable declaration white space is not allowed.

***Q.2] b) write a program which calculates factorial of given number. (02-marks for logic, 02-marks for syntax)( Note: Above program can be also written using recursion).***

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,i;
    long int fact=1;
    clrscr();
    printf("enter value of n:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        fact=fact*i;
    }
    printf("factorial of number is %ld",fact);
    getch();
}
```



***Q.2] c) Explain nested if-else statement execution with its flowchart. (01-mark for syntax, 01-mark for logic, 02-marks for flowchart)***

Nested if else statement:

When a series of decisions are involved, we may have to use more than one if...else statement in nested form as shown below:

Syntax:

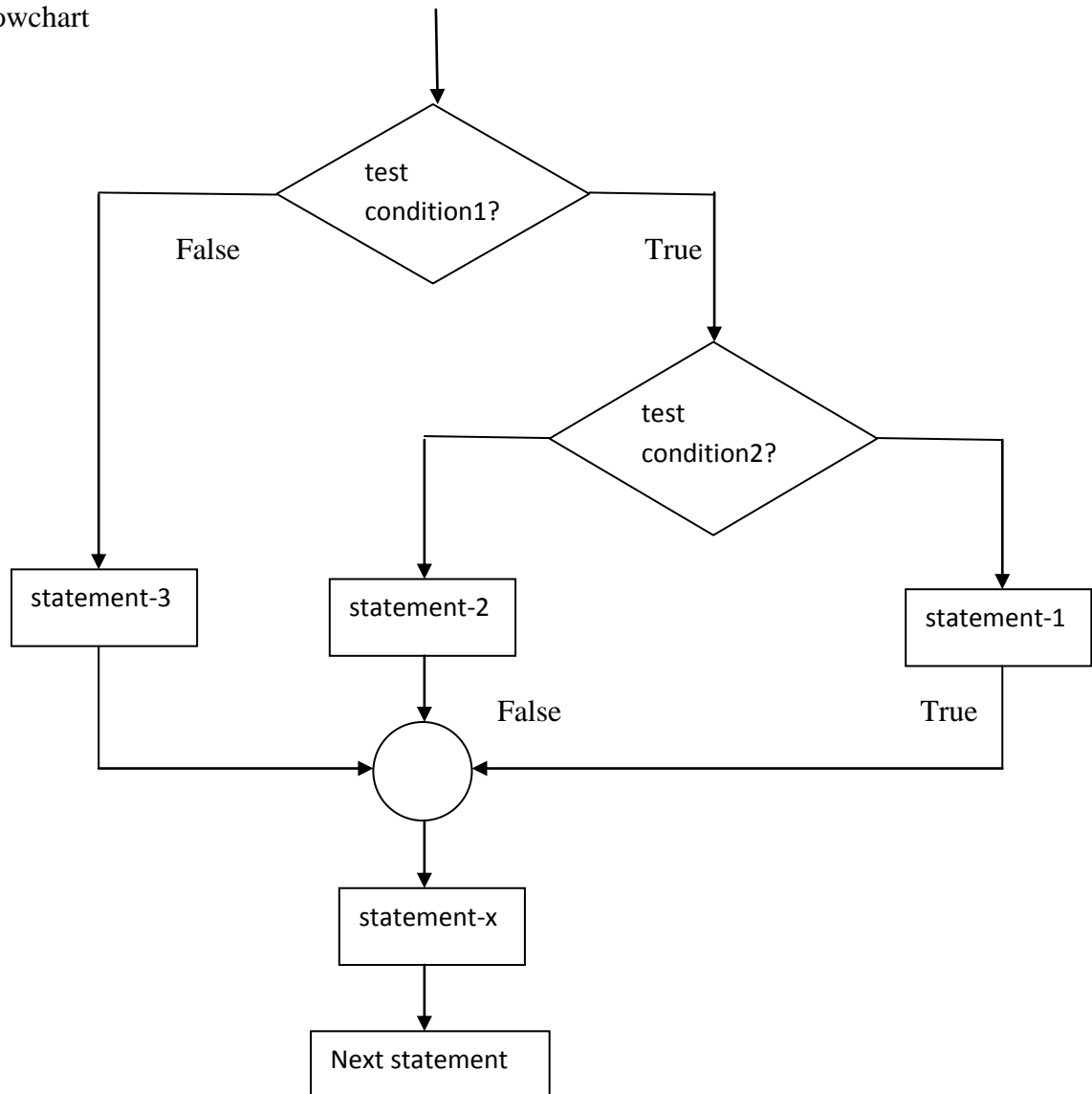
```
if(test condition1)
{
    if(test condition2)
    {
        statement-1;
    }
    else
    {
        statement-2;
    }
}
else
{
    statement-3;
}
statement-x;
```

**Logic of execution:**

If condition1 is false, the statement-3 will be executed, otherwise it continues to perform the second test. If the condition-2 is true, the statement1 will be executed otherwise the statement-2 will be executed and the control is transferred to the statement-x.



Flowchart





**Q.2] d) Compare while and do-while loop. (01-mark each)**

While Loop	Do while loop
While loop checks the condition at the beginning of loop	Do while loop checks the condition at the end of the loop.
While loop execution is more accurate as compared to do while loop	Do while loop is not as accurate as while loop.
The false condition can be run once also	It can run false condition once because to the condition is checked at the end of loop.
The syntax is: While (condition) { --- --- }	The syntax is Do { --- --- } while condition.

**Q.2] e) Explain arrays of structures with syntax and example.(02-marks for explanation, 02-marks for syntax)**

i. Array is a collection of same type of elements. In the same manner structure is collection of same or different data types of elements. When we would like to store more than one record in the memory, we can use array of structures. This array gets stored continuously in the memory.

ii. Due to continuous storage, access to this array becomes easy.

iii. in array of structures, each element of the array represents a structure variable.

Syntax:

Struct structure\_name

{

data\_type member1;

data\_type member2;

.





.

.

data\_type member n;

}struct\_variable[subscript number];

Example

Struct marks

{

int phy;

int chem.;

int maths;

} student[3];

In the above example student is the structure variable. Student[3] is an array which can store three records of student details. Each record will contain marks of physics, chemistry and maths.

***Q.2] f) What is function? Explain the need of function with example.(01-mark for definition, 02-marks for need, 01-mark for example)(Note: any other example can be considered)***

A function is a self contained block of statements which is used to perform a specific task.

Need:

1. It is possible to code any program utilizing only main function but it leads to a numbr of problem.
2. The program may become too large and complex and as a result the task of debugging, testing and maintaining becomes difficult.
3. If a program is divided into functional parts, then each part may be independently coded and later combined into a single unit.
4. These independently coded programs are called as subprograms that are much easier to understand, debug and test.

Example:

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

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

```
int add (int,int);
```

```
void main()
```

```
{
```



```
int a,b,c;
clrscr();
printf("Enter two numbers:");
scanf("%d %d", &a, &b);
c=add(a,b);
printf("\nsum=%d", c);
getch();
}

int add(int x, int y)
{
int sum;
sum=x+y;
return (sum);
}
```

Note: Any other example can be considered.

***Q.3] a)(02-marks for each case)***

The following is a segment of a program.

X=1;

Y=1;

if (n>0)

X=X+1

Else

Y=Y-1;

Printf("%d%d", X, Y);

What will be the value of X and Y if n assumes a value of 1 and if n=0?

First case

if n=1 then

X=2

Y=1



Second case

if  $n=0$  then

$X=1$

$Y=0$

***Q.3] b) (02-marks for logic, 02-marks for syntax)***

Write a program to display table upto 30. Use for loop.

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

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

```
void main()
```

```
{
```

```
int n1,n2,result;
```

```
clrscr();
```

```
for(n1=1;n1<=30;n1++)
```

```
{
```

```
printf("\n");
```

```
for(n2=1;n2<=10;n2++)
```

```
{
```

```
result=n1*n2;
```

```
printf("\n%d*%d=%d",n1,n2,result);
```

```
}
```

```
}
```

```
getch();
```

```
}
```

***Q.3] c) (02-marks for logic, 02-marks for syntax)***

Write a program to find largest number from given array.



```
#include<stdio.h>

#include<conio.h>

void main()

{

int num[5],i,largest;

clrscr();

for(i=0;i<5;i++)

{

printf("\nEnter number=");

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

}

largest=num[0];

for(i=0;i<5;i++)

{

if(num[i]>largest)

{

largest=num[i];

}

}

printf("\n%d is largest number",largest);

getch();

}
```

***Q.3] d) (04-marks for explanation)***

Explain the need for array variables:

A variable of fundamental data types can store only one value at any given time. But in many applications we need to handle large amount of data in terms of reading, processing and printing. To



process such large amounts of data, we need a powerful data type that would facilitate efficient storing, accessing and manipulation of data items. C supports a derived data type known as array.

Array variable can be used to store collection of same type of elements stored in contiguous location. Array can be used to represent list of numbers or a list of names. Some examples where array can be used:

- List of employees in an organization.
- Test scores of a class of students.
- List of customers and their telephone numbers.

We can use arrays to represent not only simple lists of values but also tables of data in two, three or more dimensions.

Example:

**Q 3 e) 02-marks for logic, 02-marks for syntax)**

Write a program to accept 10 numbers in an array. Sort the elements in ascending order.

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

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

```
void main()
```

```
{
```

```
int arr[10],i,j,temp;
```

```
clrscr();
```

```
printf("Enter array elements:");
```

```
for(i=0;i<10;i++)
```

```
{
```

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

```
}
```

```
printf("\n\nArray elements are:");
```

```
for(i=0;i<10;i++)
```

```
{
```

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

```
}
```



```
for(j=0;j<10;j++)
{
    for(i=0;i<10;i++)
    {
        if(arr[i+1]<arr[i])
        {
            temp=arr[i];
            arr[i]=arr[i+1];
            arr[i+1]=temp;
        }
    }
}

printf("\n\nArray elements in ascending order are:");

for(i=0;i<10;i++)
{
    printf("\n%d",arr[i]);
}

getch();
}
```

***Q.3] f)(04-marks for explanation)***

Explain how array elements are accessed using pointer.

when an array is declared, the compiler allocates a base address and sufficient amount of storage to contain all the elements of the array in contiguous memory locations. The base address is the location of the first element (index 0) of the array. The compiler also defines the array name as a constant pointer to the first element. Suppose we declare an array x as an array as follows:

```
int x[5] = {1,2,3,4,5};
```



suppose the base address of x is 1000 and assuming that each integer requires two bytes, the five elements will be stored as follows:

Elements     $\longrightarrow$     x[0]            x[1]            x[2]            x[3]            x[4]

Value     $\longrightarrow$ 

1	2	3	4	5
---	---	---	---	---

Address     $\longrightarrow$     1000            1002            1004            1006            1008

The name x is defined as a constant pointer pointing to the first element, x[0] and therefore the value of x is 1000, the location where x[0] is stored. That is ,

$$x = \&x[0] = 1000$$

if we declare p as an integer pointer, then we can make the pointer p to point to the array x by the following assignment:

p=x;

this is equivalent to    p=&x[0];

***Q.4] a) (02-marks for use, 01-mark for precedence, 01-mark for associativity)***

State the use of increment and decrement operator and give its precedence and associativity.

The operator ++ adds 1 to the operand, while –subtracts 1. Both are unary operators. We use the increment and decrement statements in for and while loops extensively.

Precedence: 2

Associativity: Right to left.

***Q.4] b) (02-marks for logic, 02-marks for syntax)***

Print the following pattern

1

2        3

4        5        6

#include<stdio.h>



```
#include<conio.h>

void main()

{

int i=0,j=0,cnt=1;

clrscr();

for(i=1;i<=3;i++)

{

printf("\n\n");

for(j=1;j<=i;j++)

{

printf("\t%d",cnt);

cnt++;

}

}

getch();

}
```

***Q.4] c) (1 ½-marks for declaration, 1 ½-marks for initialization, 01-mark for example)(Note: any other example can be considered)***

Describe a method of declaring and initializing of two dimensional array with example.

Declaration of two dimensional array:

Syntax

datatype arrayname[row-size][column-size];

e.g. int n[3][3];

in the above example, we have declared a as a integer, it will accept only integer values. 3 is the size of rows and columns.

Initialization of two dimensional array.





```
int num[3][3] = { 1,2,3,4,5,6,7,8,9};
```

OR

```
int num[3][3] = { { 1,2,3},  
                  {4,5,6};  
                  {7,8,9}  
                };
```

In the above example, there will be 3 rows and 3 columns. The one pair of brackets represents values for one row.

***Q.4] d) (02-marks for logic, 02-marks for syntax)***

Write a program to accept a string and display the length of it without using standard library function.

```
#include<stdio.h>  
  
#include<conio.h>  
  
#include<string.h>  
  
void main()  
{  
    int i=0;  
    char str[10];  
    clrscr();  
    puts("Enter string");  
    gets(str);  
    while(str[i]!='\0')  
    {  
        i++;  
    }  
    printf("length=%d",i);
```



```
getch();
```

```
}
```

***Q.4] e) (02-marks for arithmetic, 02-marks for example)***

Explain the pointer arithmetic's with example.

Pointer arithmetic is concerned with the arithmetic operations with the pointers. Like basic operation +, -, \*, / can be done using pointer notation. Some of the following operations are possible:

e.g.  $\text{add} = *p1 + *p2$

$y = *p1 - *p2$

$x = *p1 / *p2$

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

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

```
void main()
```

```
{
```

```
int a=10,b=2,sum,mul;
```

```
int *p1,*p2;
```

```
clrscr();
```

```
p1=&a;
```

```
p2=&b;
```

```
sum=*p1+*p2;
```

```
mul=*p1**p2;
```

```
printf("\nAddition=%d\nMultiplication=%d",sum,mul);
```

```
getch();
```

```
}
```

***Q.4] f) (02-marks for logic, 02-marks for syntax)***

Write a program for exchanging values of two variables using call by reference.



```
#include<stdio.h>

#include<conio.h>

void swap(int *,int *);

void main()

{

int a=0,b=0;

clrscr();

printf("\n Enter two nos=");

scanf("%d%d",&a,&b);

swap(&a,&b);

printf("\na=%d b=%d",a,b);

getch();

}

void swap(int *x,int *y)

{

int temp;

temp=*x;

*x=*y;

*y=temp;

}
```

***Q 5 ) a)(logic 2 marks, syntax 2 marks) Write a program to demonstrate all possible formatting specifiers.***

```
#include<stdio.h>

#include<conio.h>

void main()

{
```



```
int account_no;

float intrest=0.0;

char account_type;

char name[20];

double deposit;

clrscr();

printf("\n enter account number & name");

scanf("%d %s", account_no,name);

printf("enter deposit & interest");

scanf("%ld %g",&deposit ,&intrest);

printf("account_no:%d\tdeposit:%ld\tintrest:%f",account_no,name,deposit,intrest);

getch();

}
```

O/P:

Enter account number & name    102    abc

Enter deposit & interest            25000    10.5

Account\_no:102        name:abc        deposit:25000        interest=10.5

***Q.5] b) Write any four arithmetic operator & four logical operator with example.(02-marks for arithmetic operator -02-marks for logical operator) Note: Consider any example related to arithmetic operator & logical operator***

Ans: Arithmetic operators are popularly used to do basic operation like addition, subtraction, multiplication & division .

Number	Operator	Meaning
1	+	Addition
2	-	Subtraction
3	*	Multiplication
4	/	Division
5	%	Modulus



Example:1) Modulus operator:  $10 \% 2$  is 0 i.e. 10 is divisible by 2 so that remainder is 0 & quotient will be 5.

2) Addition operator:

$10+2$  is 15

Logical Operator

Number	Operator	Meaning
1	&&	AND-True if all conditions are true
2		OR- True if any one or all conditions are true
3	!	NOT- Negation

Example:

```
if((time>=0)&&(time<12)
```

```
printf("Good Morning");
```

```
else
```

```
if((time >=12 ) && (time <18))
```

```
printf("good Afternoon");
```

```
else
```

```
if((time >=18) && (time < 24)
```

```
printf("Good Evening");
```

```
else printf("Time is out of range");
```

***Q.5] c) Write a program 'c' to reverse the entered integer number as input through keyboard.***

***(02-marks for logic –02-marks for syntax)***

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

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

```
void main()
```

```
{
```

```
int n, reverse = 0;
```

```
printf("Enter a number to reverse\n");
```

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

```
while (n != 0)
```

```
{
```

```
reverse = reverse * 10;
```



```
reverse = reverse + n%10;
n = n/10;
}
```

```
printf("Reverse of entered number is = %d\n", reverse);
```

```
getch();
}
```

o/p: Enter a number 23

Reverse of entered number is= 32

***Q.5] d) Give any four forms of if & if-else statement.(1 mark for each form)***

1. If statement : if statement can be used to check whether condition is true or not.

Syntax:

If(condition)

{

----- statements

-----

}

2. If else loop: It includes two parts one is called true part & another is called as false part. The statements enclosed in if brackets would get execute if the condition if the condition is true otherwise if condition becomes false then else statements will get execute .

Syntax:

If(condition)

{

----- statement

-----

}

Else

{

----- statements

-----

}

3. if Else if Ladder: If else if ladder is used when there are multiple conditions to be checked.

Syntax:

if(condition 1)

{



```
----
----
}
else if (condition 2)
{
----
----

}

else

{
----
----
}
```

4. Nested if: When multiple conditions are introduced in a particular sequence,

Syntax:

If(condition 1)

{

If(condition 2)

{

---

---

}

Else

{

----

----

}

}

Else

{

----

----

}

***Q.5] e) Explain global & local variable. Write 'c' program to explain global & local variables.(02-marks for Explanation, 02-marks for Program )***

**Global Variable:** Global variables are external variable which are declared above the main(). These variables are accessible to all the functions in the program. The lifetime & scope of these variables is throughout the program. Any function can use this value & can manipulate it whenever required.

**Local Variable:** Local variables are declared in the function in which they get utilized.

They get created when function is called & gets destroyed when function execution gets over.



C program:

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

```
// Global variables
int A;
int B;

int Add()
{
    return A + B;
}

void main()
{
    int answer; // Local variable
    A = 5;
    B = 7;
    answer = Add();
    printf("%d\n",answer);
    getch();
}
```

***Q.5] f) Write a program to define a structure employee with members emp\_name, emp\_id, emp\_salary . Accept data for any one employee & display it.( 02-marks for Logic -02-marks for syntax)***

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

```
void main()
{
struct emp
{

int emp_id;
char emp_name[10];
float emp_sal;
}e;
clrscr();
printf("\n enter emp_id");
scanf("\n %d",&e.emp_id);
printf("\n enter emp_name");
scanf("\n%s",&e.emp_name);
printf("\n enter emp_sal");
```





```
scanf("\n%f",&e.emp_sal);
printf("\nemp_id=%d\temp_name=%s\temp_sal=%f",e.emp_id,e.emp_name,e.emp_sal);

getch();
}
```

O/P:

Enter emp\_id 11

Enter emp\_name xyz

Enter emp\_sal 200.50

Emp\_id=11      emp\_name=xyz      emp\_sal=200.50

***Q.6]a) Describe different data type support by c lang.(01-mark for types-03-marks for explanation of any three data types)***

Data Types are used to specify type of data. The data is collection facts, numbers. For accepting any type of data in the variable required to specify the data type with its variable.

Data Types are the specification provided with the variable or function name. These data types are categorized into three types as follows:

1. Primary or fundamental data types
2. Derived data types.
3. User defined data types.

The following five data type fall under primary data types:

1. Character-char
2. Integer-int
3. Float-float
4. Double-double

Character data type is used to store single character or number at a time.

memory space required for character type of data is 8 bytes. Ranges of values is -128 to 127

Integer allows to store integer values only. No decimal points are allowed. Integer takes 16 bytes. Integer can be positive or negative. -32768 to 32767.

Float type stores only floating point values. The values should be with the decimal places. Float takes 32 bytes. Range of values is 3.4e-38 to 3.4e+38



Double type stores the double value than float. Double takes 64 bytes of memory.

The range for storing double value is  $1.7e-308$  to  $1.7e+308$ .

**Q.6] b) Explain switch statement with the flowchart & example. ( 02-marks for explanation & flowchart, 02-marks for example)**

Switch statement accepts value & tests it for various conditions. If all conditions are false then the default statement will get execute.

The diagram shows pictorial representation of execution of switch case loop. The conditions are checked by one by one.

The statement of the case would be execute if the condition is true. Else control pass to the next case.

The different conditions in switch are called as cases. These cases are break using break keyword. If any one of the case is true then statements written in the same case would get execute & the case will get due to break keyword.

example: Accept character form user & check character is vowel or not using switch statement.

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

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

```
void main()
```

```
{
```

```
char code;
```

```
clrscr();
```

```
printf("\n Enter character:");
```

```
scanf("%c",&code);
```


```
switch(code)            //passing code to check the value
```

```
{
```

```
case 'a':
```

```
case 'e':
```

```
case 'i':
```

```
 // all cases a,e,i,o u are single character so enclosed in
```



single quotes

```
case 'o':
```

```
case 'u':
```

```
printf("%c is vowel",code);
```

```
break;
```

```
default:
```

```
printf("the %c is not vowel",code);
```

```
break;
```

```
}
```

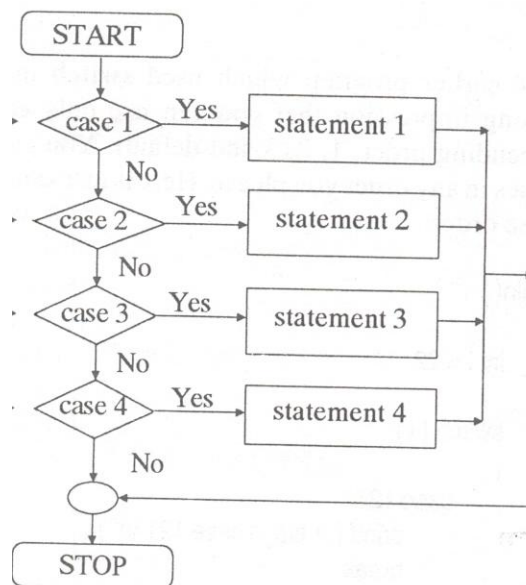
```
getch();
```

```
}
```

O/P:

Enter character: s

s is not vowel





**Q.6] c) Describe the string function strlen(), strcpy() with example.**

**(02-marks for explanation with example of strlen()-02-marks for strcpy())**

1. strlen():- It can be used to get the length of the string. It uses string argument which is name of character array. It returns an integer showing no. of characters from the string excluding last '\0' (null character).

Ex: int n;

```
n=strlen("abc");
```

then n=3.

2. strcpy() :- This function accepts two string arguments and as a result contents of string2 are copied in string1.

Ex. char s1[20]={ "ABC"};

```
char s2[8]={ "Company "};
```

```
strcpy(s1,s2)
```

Output: s1= "Company"

s2= "Company "

Note: (Any other example can be considered. )

**Q.6] d) Difference between call by value & call by reference (01-mark each)**

Call by value	Call by reference
1.When functions are called by passing values then they are called as call by value.	1.When functions are called by passing addresses then they are called as call by reference.
2.It is risky to call function by values because values may get conflict or interchange in large program.	2.There is no risk when functions are called by passing the address of variable though the program is large or complicated.
3.Call by values wastes the memory	3.call by reference does not waste the memory as new variables are not created.
4.Call by value is less efficient than call by reference.	4.Call by reference is more efficient then call by value.

**Q.6] e) Write a program C using pointers to compute the sum of elements stored in array abc[ ].**



*(02-marks for logic-02-marks for syntax)*

```
#include<stdio.h>

#include<conio.h>

void main()

{

int *ptr,sum=0,abc[4],i=0;

for(i=0;i<=3;i++)

{

printf("\n enter array elements");

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

ptr=&abc[i];

sum=sum+(*ptr);

}

printf("\n Sum of array elements=%d",sum);

getch();

}
```

O/P

enter array elements 10

enter array elements 20

enter array elements 30

enter array elements 40

Sum of array elements=100

***Q.6] f) Explain the meaning of following statement with reference to pointer.( 02-marks for first statement-01-mark for second statement, 01-mark for third statement)***

```
int *ptr m=10
```



ptr=m;

ptr=&M

1) Int \*ptr, m=10 The meaning of this statement is that ptr is a pointer and m is integer variable initialized as 10

2) ptr=m assigning value of m i.e. 10 to the pointer ptr.

3) Meaning of ptr=&m means assigning address of m to the pointer ptr.

-----\*\*\*\*\*-----