Assingment - 1.

Q1 Using bitwise operator AND, write a program in C to test whether a given number is even or odd.

```
#include <stdio.h>
int main()
{.
        int num;
        printf (" Enter any number :  ");
        scanf ("%d", &num);

        if(num & 1)
        {
            printf("%d is odd. ", num);
        }.
        else
        {
            printf("%d is even.", num);
        }.
        return 0;
}
```

→ here if(num & 1) is equivalent to if((num&1)==1).

→ The Least Significant Bit of an odd number is set(1). To check wether a number is even or odd we need to figure out if LSB is set or not.

as for 12 (decimal) 13.

00001100(In Binary) 00001111

FOR EDUCATIONAL USE

→ ∴ we use Bitwise AND & operator to check whether a bit is set or not.

→ On performing num & 1 it returns LSB of num. If LSB is 1 then the given number is odd otherwise even.

Q2. What are the pointers? Write a program to show how a pointer to an array of integers can be passed to a function to ~~sort~~ short 'n' numbers present in an array.

→ Pointers in C Language is a variable ~~tot~~ that stores/points the address of another variable. A pointer in C is used to allocate memory dynamically i.e. at run time.
Can be of any data type such as int, float, char, double, short, etc.

```
→ #include <stdio.h>
  void Sort (int n , int*ptr)
  {
     Int i,j,t;
     for( i=0; i<n; i++)
     {
        for( j= i+1 ; j<n; j++)
        {
           if ( *(ptr+j) < *(ptr +i))
           {
              t = *(ptr+i);
              *(ptr+i) = *(ptr+j);
              *(ptr+j) = t;
           }
        }
     }
```

```
        for( i=0 ; i<n; i++)
          {
               printf("%d" , *(ptr + i)) ;.
          }.
   }.

int  main()
{
        int   n ~~[illegible]~~ ;
        int   arr[illegible][n] ;.
        printf(" Enter the  no of elemets");
        scanf ( " %d", ~~&~~& n);
        printf(" Enter the   %d elmets of array" , n);
        ~~Scan~~
         for ( int  i=0 ; i <n ; i++)
            {
                  Scanf ( "%d" , & arr[i]);
            }.
         Sort (n ,arr) ;   return 0 ;
}.
```

Q3. Define calling function? What is 'call by value' parameter passing in c program? How it is differ from call by reference.

→ A calling function invokes the called function, by supplying the required arguments, if necessary, to the called function. The calling function receives the return value from the called if there is one.

→ The call by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function.

by default, C programming uses call by value to pass argumets. In general, it means the code withen a function can not often. the arguments used to call the function

→. In call by value A copy of the value is passed into the function, and the changes made inside the function is limited to the function only. The values of the actual parameters do not change by changing the pormal parameters.

whereas, In call by reference An address of value. is passed into the function, and changes made inside the function validate outside of the function also. The values of the actual parasameters do ~~so~~ change by changing

The formal parameters.

Q4. Describe the basic structure of c program.

→

To write a c program, we first create function and then put them together. A C program may contain one or more sections, They are illustrated below.

Documentation section. [used for comments]

~~Global declaration section.~~

Link section → | # include <stdio.h>.

Defination section → | Void fun () ;

Global declaration section | → int a=10 ;
[Variable used in ~~one~~ more than one function].

main() → | Void main()

```
main()                      | Void main()
{                           | {
   Declaration part         | clrscr();
   Executable part          | printf("a value inside main() : %d", a);
}                           | fun();
                            | }.
```

Subprogram section. → [user-defined function] Function 1 function 2 function 3 : : function n.	Void fun () { printf (" /n a value inside fun() : %d ", a); }.

Q5. Explain the difference between pre-increment and post-increment [i.e ++X, X++] operators using an appropriate example.

→ Pre increment operator is used to increment variable value by 1 before assigning the value to the variable.

→ Post increment operator is used to increment variable value by 1 after assigining the value to the variable.

```
# include <stdio.h>
Void Main()
 {.
     int I = 42;.
     int a,b ;
     a= I++ ;   // post -increment , a will contain 42
     b= ++I;    // pre - increment , b will contain 43
  }.
```