1. Write a program in C to count the frequency of each element of an array.

```
#include <stdio.h>

int main()

{

   int arr1[100], fr1[100];

   int n, i, j, ctr;

   printf("\n\nCount frequency of each element of an array:\n");

   printf("Input the number of elements to be stored in the array :");

   scanf("%d",&n);

   printf("Input %d elements in the array :\n",n);

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

         {

            printf("element - %d : ",i);

            scanf("%d",&arr1[i]);

                 fr1[i] = -1;

         }

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

   {

      ctr = 1;

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

      {
            if(arr1[i]==arr1[j])
            {
                ctr++;
                fr1[j] = 0;
            }
        }

        if(fr1[i]!=0)
        {
            fr1[i] = ctr;
        }
    }
    printf("\nThe frequency of all elements of array : \n");
    for(i=0; i<n; i++)
    {
        if(fr1[i]!=0)
        {
            printf("%d occurs %d times\n", arr1[i], fr1[i]);
        }
    }
return(0);
```

```
}
```

3. Write a C program to show that pointer of any data type occupies same space.

```
#include<stdio.h>

int main() {

 int x= 38, *ptrx = &x;

 double y = 5.85, *ptry = &y;

 char ch = 'H' , *ptrc = &ch;

 char Name[] = "Ramayana";

 char *ptrname = Name;

 /*& is not used before Name because Name is itself const pointer to the string.*/

  printf ("Size of ptrx = %d bytes\n", sizeof(ptrx));

 printf ("Size of ptry = %d bytes\n", sizeof(ptry));

 printf ("Size of ptrc = %d bytes\n", sizeof(ptrc));

 printf ("Size of ptrname = %d bytes\n" , sizeof(ptrname));

 printf ("Size of x = %d bytes\n" , sizeof(x));

 printf ("Size of y = %d bytes\n", sizeof(y));

 printf ("Size of ch= %d bytes\n" , sizeof(ch));

 printf ("Size of Name= %d bytes\n", sizeof(Name));

 return(0);

}
```

Functions

1. Names of parameters in a function prototype have to match the names given in the function definition. TRUE/**FALSE**

2. Write a function named int zeroCheck(int a, int b, int c); that is given three integers, and returns 1 if any of the integers is 0, otherwise it returns 0.

```
int zeroCheck(int a, int b,int c)
{
        if(a==0 || b==0 || c==0)
                return 1;
        else
                return 0;
}
```

3. What is the output for the following code:

```
#include <stdio.h>
int what(int a, int n)
{
        if(n == 0)
                return 1;
        else if(n % 2)
                return a * what(a * a, n / 2);
        else
                return what(a * a, n / 2);
}
int main()
{
        int a = 3, b = 5;
        printf("%d\n", what(a, b));
}
```

Output:
243

4. Which of the following would be valid prototypes for a function that returns nothing and has one double parameter?

a. **void f(double x);**
b. **void f(double);**
c. void f(x);
d. f(double x);

5. Write a program to find the number of digits in an interger using recursion.
   int length(int n);
   length(892) will return 3
   length(3452) will return 4.

```
int len(int n)
{
        if(n==0)
                return 0;
        else
                return 1+len(n/10);
}
```

1. Consider following array

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

Assume the base address of array p=1000.

find the address of P[2][1]?(2D arrays follows Row Major ordering)

Answer: p[2][1]=1000+((2*3)+1)*4=1000+24=1024

7.Write a program in C to find the row with maximum number of 1s using Functions

The given 2D array

0 1 0 1 1

1 1 1 1 1

1 0 0 1 0

0 0 0 0 0

1 0 0 0 1

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

#define R 5

#define C 5

 int getFirstOccur(int arr1[], int l, int h)

{

if(h >= l)

{

    int mid = l + (h - l)/2;

    if ( ( mid == 0 || arr1[mid-1] == 0) && arr1[mid] == 1)

    return mid;

    else if (arr1[mid] == 0)

    return getFirstOccur(arr1, (mid + 1), h);

    else

    return getFirstOccur(arr1, l, (mid -1));
}
return -1;
}
int findRowMaxOne(int arr2d[R][C])
{
   int max_row_index = 0, max = -1;
   int i, index;
   for (i = 0; i < R; i++)
   {
   index = getFirstOccur (arr2d[i], 0, C-1);
   if (index != -1 && C-index > max)
   {
      max = C - index;
      max_row_index = i;
   }
   }
   return max_row_index;
}

int main()
{
   int arr2d[R][C] = { {0, 1, 0, 1,1},
                  {1, 1, 1, 1, 1},
                  {1, 0, 0, 1, 0},
                  {0, 0, 0, 0, 0},
                  {1, 0, 0, 0, 1}
   };
   int i,j;
   printf("The given 2D array is :  \n");
           for(i = 0; i < R; i++)
           {
           for(j=0; j<C ; j++)
           {
           printf("%d  ", arr2d[i][j]);
           }
printf("\n");
    }
 printf("The index of row with maximum 1s is:  %d " , findRowMaxOne(arr2d));
    return 0;
}
```

2. Where does global, static, local, register variables and C Program instructions get stored?

Global , static, local : In main memory
Register variable: In registers
C program : In main memory

3. Identify the error in the below code and explain

```
#include<stdio.h>
main ()
{
extern int i; i=20;
printf("%d",i);
 }
```

Answer: Linker Error : Undefined symbol '_i'
Explanation: extern storage class in the following declaration, extern int i; specifies to the compiler that the memory for i is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name i is available in any other program with memory space allocated for it. Hence a linker error has occurred .

7. What is the output of the program? Explain your answer

```
void myshow();

int main()
{
        myshow();
        myshow();
        myshow();
}

void myshow()
```

```
{
        static int k = 20;
        printf("%d ", k);
        k++;
}
```

Ans - 20 21 22

Variable of type static holds its value until the end of program execution .Static variables do not initialize again and again with function calls

1. **What is Enum in C? Give an example.**

Enumeration (or enum) is a way of creating user defined data type in C. It is mainly used to assign names to integral constants. Enumeration data type consists of named integer constants as a list. It start with 0 (zero) by default and value is incremented by 1 for the sequential identifiers in the list.

Syntax

enum identifier {enumerator-list};

Egs –

```
enum month { Jan, Feb, Mar };
```

Jan,Feb and Mar variables will be assigned to 0, 1 and 2 respectively by default

```
enum month { Jan = 1, Feb, Mar };
```

Feb and Mar variables will be assigned to 2 and 3 respectively by default

3. **What will be the output of the following C code?**

```
#include <stdio.h>
enum example {a = 1, b, c};
enum example example1 = 2;
enum example answer()
{
    return example1;
}
 int main()
{
   (answer() == a)? printf("yes"): printf("no");
   return 0;
 }
```

Ans- no

Explanation: In the code shown above, the value of example1 is returned by the function answer. The ternary statement prints yes if this value is equal to that of ‘a’ and no if the value is not equal to that of ‘a’. Since the value of ‘a’ is 1 and that returned by the function is 2, therefore no is printed.

6. **What is the benefit of using an enum rather than a #define constant?**

The use of an enumeration constant (enum) has many advantages over using the traditional symbolic constant style of #define. These advantages include a lower maintenance requirement, improved program readability, and better debugging capability.

1)The first advantage is that enumerated constants are generated automatically by the compiler. Conversely, symbolic constants must be manually assigned values by the programmer.

2) Another advantage of using the enumeration constant method is that the programs are more readable and thus can be understood better by others.

3) A third advantage to using enumeration constants is that some symbolic debuggers can print the value of an enumeration constant. Conversely, most symbolic debuggers cannot print the value of a symbolic constant. This can be an enormous help in debugging user program, because if your program is stopped at a line that uses an enum, it can simply inspect that constant and instantly know its value. On the other hand, because most debuggers cannot print #define values, user would most likely have to search for that value by manually looking it up in a header file.