Section – C

Attempt any three questions out of the following five questions. Each question carries 15 marks.

Q9. What kind of file accesses is possible with 'C' programming? What are the different modes that a file in 'C' can be handled?

Ans. A file is a collection of bytes stored on a secondary storage device which is generally a disk of some kind. There are two kind of files in C programming :

1. Text file
2. Binary file

1. Text File : A text file can be a stream of character that a computer can process sequentially. It is not only processed sequentially but also in forward direction. A text stream is a sequence of characters. Standard 'C' allows a text stream to be organised into lines terminated by a new line character (new line character is optional).

2. Binary File : A binary file is no different to a text file. It is a collection of bytes. In 'C' programming language a byte and a character are equivalent. Hence a binary file is also referred to as a character stream but there are two essential differences :

(a) No special processing of the data occurs and each byte of data is transferred to or from the disk unprocessed.

(b) 'C' programming language places no constructs on the file and it may be read from or written to, in any manner chosen by the programmer.

To store or to retrieve the data to or from a file, the file should be opened in the beginning of the program. To open a file the function fopen( ) is used. This function returns a pointer to a file.

file_pointer = fopen("filename", "mode");

Mode	Meaning
r	For reading
w	For writing
a	Append to a text file
rb	Open binary file for reading
wb	Open binary file for writing
Ab	Append to binary file
R+	Open a text file to read/write
W+	Create a text file to read/write
A+	Append a text file to read/write
rb+	Open a binary file to read/write
wb+	Open binary file to read/write
ab+	Append to binary file to read/write

Q10. What are the uses of Malloc () and Calloc () function in 'C'? In which Header file these two functions are defined? What is the full syntax of these two functions?

Ans. The Malloc( ) Function :

The function malloc is used to allocate a certain amount of memory during the execution of a program. The malloc function will request a block of memory from the heap. If the request is granted, the operating system will reserve the requested amount of memory.

The malloc( ) function allocates a block of memory in bytes. The user should explicitly give the block size it requires for the use. The malloc( ) function is like a request to the RAM of the system to allocate memory, if the request is granted (i.e., the malloc function stays successful in allocating memory), returns a pointer to the first block of that memory. The type of the pointer it returns is void, which means that we can assign it any type of pointer. However, if the malloc( ) function fails to allocate the required amount of memory, it returns a NULL.

The syntax of this function is as follows :

```
malloc(number of elements * size of each element);
```

For example :

```
int * ptr ;
ptr = malloc(10 * sizeof(int) )
```

where sizeof represents the size of memory required in bytes.

But as already told that the function malloc( ) returns a void pointer so a cast operator is required to change the returned pointer type according to our need, the above declaration would take the following form :

```
ptr_var = (type_cast*)malloc(size)
```

where ptr_var the name of pointer that holds the starting address of allocated memory block, type_cast is the data type into which the returned pointer (or type void) is to be converted, and size specifies the size of allocated memory block in bytes.

For example :

```
int *ptr;
ptr = (int*)malloc(5 *sizeof(int) );
if(ptr = = NULL)
{  printf("The required amount of memory is not available");
   getch( );
   exit(0);
}
```

The Calloc( ) Function :

This function works exactly similar to malloc( ) function except for the fact that it needs two arguments as against one argument required by malloc().

For example :

```
int *ptr;
ptr = (int*)calloc(10, 2);
```

Here 2 specifies the size of data type in byte for which we want the allocation to be made, which in the case is 2 for integers. And 10 specify the number of elements for which allocation is to be made. The argument passed to function malloc was (n*10), it is a single argument because multiple arguments are always separated by commas. The argument (n * 10) has no commas in between hence it is a single argument, though not a simple one but an expression. Returning to the above declaration – after the execution of the above statement a memory block of 20 bytes is allocated to the requesting program and the address of first block is assigned to pointer ptr. Another minor difference is that the memory allocated by malloc( ) function contains garbage values, while memory allocated by calloc( ) function contains all zeros.

Note : The **malloc( )** and **calloc( )** functions are available in header file **alloc.h** or **stdlib.h**.

Q11. Write a 'C' program to create data for 50 students (roll, name, mark 1, mark 2, mark 3, term mark) using structure and then find the total marks for each student and average marks of all the students?

Ans.

```
#include <stdio.h>
#include <conio.h>
void main( )
{
 struct student
 {
  int rollno;
  char name[30];
  int m1, m2, m3, tm;
  int total
 }
struct student s[50];
int i, n;
int sum = 0, average;
clrscr( );
for(i = 0; i < 50; i++)
{
    printf("\n enter the rollno \n");
    scanf("%d", &s[i].rollno);
    printf("\n enter the name \n");
    scanf("%s", s[i].name);
    printf("\n enter the mark m1=");
    scanf("%d", &s[i].m1);
    printf("\n enter the mark m2=");
    scanf("%d", &s[i].m2);
    printf("\n enter the mark m3=");
    scanf("%d", &s[i].m3);
    printf("\n enter the mark tm=");
    scanf("%d", &s[i].tm);
 s[i].total = s[i].m1 + s[i].m2 + s[i].m3 + s[i].tm;
 }
 for(i = 0; i < 50; i++)
 {
     sum = sum + s[i].total;
     average = sum/50;
 printf("Total marks of student = %d", s[i].total);
 }
 printf("Average marks = %d", average);
 getch( );
}
```

Q12(a). Write a 'C' program to count the number of vowels in a given string.

Ans.

```
#include <stdio.h>
int count_vowels(char[ ]);
int check_vowel(char);
main( )
{ char array[100];
  int c;
  printf("Enter a string \n");
  gets(array);
  c = count_vowels(array);
  printf("Number of vowels : %d \n", c);
  return 0;
}
int count_vowels(char a[ ])
{
  int count = 0, c = 0, flag;
  char d;
  do
  {      d = a[c];
         flag = check_vowel(d);
         if(flag = = 1)
         count + +;
         c++ ;
  } while (d! = '\0');
   return count ;
}
int check_vowel(char a)
```

```
{
 if(a > = 'A' && a < = 'Z')
 a = a + 'a' – 'A';        /* converting to lower case */
 if(a = = 'a' || a = = 'e' || a = = 'i' || a = = 'o' || a = = 'u')
  return1 ;
  return 0 ;
}
```