How are they return :

Each array element is referred to by specifying the array name followed by one or more subscripts with each subscript enclosed in square brackets. **C program to store the elements in an array and copy to another array and print it :**

```
#include <stdio.h>
void main( )
{
    int a[5], b[5];
    clrscr( )
    printf("enter the elements of an array:");
    for(i = 0; i < 5; i++)
    scanf("%d", & a[i]);
    for(i = 0; i<5; i++)
    b[i] = a[i];
    for(i = 0, i<5; i++)
    printf("%d", b[i]);
    getch( );
}
```

Here, a & b are the array names & '5' in bracket [] after the array names is known as subscript.

Restrictions :

1. Each subscript must be expressed as a non-negative integer. It accepts only positive value that is greater than '0'.
2. An incorrect or valid subscript may cause unexpected results.
3. There is no bound checking for the subscript in C.

Q3. What is an execution error? Differentiate it from syntactic error. Give examples.

Ans. Execution errors :

Execution errors are those errors which are caused by incorrect usage of programming logic. An execution error occurs when the program comes up to something that it cannot handle because it does not have the code for it.

Execution errors could also be buffer overruns, out of memory exception etc.

For example : If we have a division statement in our program. If for some reason the denominator becomes zero then we will get a runtime error (something like DIVISON-BY-ZERO).

Runtime errors are dynamic errors that cannot be detected by the compiler. Runtime errors can be found only when we are trying to execute our

Syntax errors :

Syntax errors are those which are caused by incorrect usage of the programming language. A syntax error is an error in how the code is written. i.e. no closing of quotes correctly, of forgetting the ';' at the end of a line of code. This is very dependent on the language that the code is written in, as to what exactly the error could be, but essentially it is malformed code. This would result in a compilation error, meaning that the program would not compile and therefore not be able to execute. All programming language compilers are designed to detect and report such errors done by the programmer.

Syntax errors are static errors that can be detected by the compiler. Syntax error can be found during compilation.

Example, Missing ';' at the end of a statement.

Q4. It is said that 'C' is a middle-level assembly language. Mention those features of 'C' which enable this description?

Ans. C programming is called middle-level-language because it is actually binding the gap between a machine level language and more conventional high-level languages. User can use 'C' language to do system programming for writing operating system as well as application programming.

C has almost all the features of low level languages like bit-level manipulation, with additional features of high level languages like elegant structure, easy to debug.

In other words, C was developed as a language with all advantages and no disadvantages of assembly language with additional significant features of modern high level languages. Lying between two categories: low level language and high level languages, 'C' is known as middle level language also.

Following are the main feature of 'C' which confirms its as middle-level language :

1. C programming language behaves as high level language through function, it gives a modular programming & breakup, increased the efficiency for resolvability.
2. C programming language supports the loco-level language i.e. assembly language.
3. C language also gives the facility to access memory through pointers.
4. It combines the elements of high level languages with the functionality of assembly

language.

5. Using inline assembly language feature in 'C', we can directly access system registers.

Q5. 'C' compiler supports many pre-processor commands. Write their names only.

Ans. The C preprocessor :

It modifies a source file before handing it over to the compiler, allowing conditional compilation with #if, def, defining constants with #define, including header files with # include and using built in macros such as _FILE_. Following are the preprocessor directives or commands to the preprocessor, that are available :

1. #include
2. #define
3. #undef
4. #if
5. #ifdef
6. #ifndef
7. #error
8. #macro operator
9. ##macro operator
10. _FILE_
11. _LINE_
12. _DATE_
13. _TIME_
14. _TIMESTAMP_

Section – B

Q6. Write a 'C' program to calculate the frequencies of different alphabets, present in a given string. The string of alphabets is to be taken as input from the keyboard.

Ans. C Program :

```
#include <stdio.h>
#include <string.h>
void main( )
{
  char string [100];
  int c = 0, count [26] = {0};
  printf("Enter a string \n");
  gets(string);
  while(string [c]! = '\0')
  {
    if(string [c] > = 'a' && string [c] < = 'z')
    count [string [c] – 'a']++;
    c++
  }
  for(c = 0; c < 26; c++)
  {
    if(count [c]! = 0)
    printf("% c's occurrence : %d times \n", c
    + 'a', count [c]);
  }
}
```

Output of program :

```
Enter a string
a quick brown fox jump over the lazy dogs
a's occurrence : 2
b's occurrence : 1
c's occurrence : 1
d's occurrence : 1
e's occurrence : 2
f's occurrence : 1
g's occurrence : 1
h's occurrence : 1
i's occurrence : 1
j's occurrence : 1
k's occurrence : 1
l's occurrence : 1
m's occurrence : 1
n's occurrence : 1
o's occurrence : 4
p's occurrence : 1
q's occurrence : 1
r's occurrence : 2
s's occurrence : 1
t's occurrence = 1
u's occurrence = 2
v's occurrence = 1
w's occurrence = 1
x's occurrence = 1
y's occurrence = 1
z's occurrence = 1
```

Q7. Develop a program to multiply two matrices with size 3×4 and 4×5. Your program should take care that no element of either matrix can be negative.

Ans. Program for matrix multiplication :

```
#include <stdio.h>
void main( )
{
  int a[10] [10], b[10] [10], m[10] [10], r1, c1,
  r2, c2, i, j, k;
  printf("enter rows & columns for first matrix:");
  scanf("%, %d", &r1, &c1);
printf("enter rows & columns for second ma-
trix :");
scanf("%d, %d", &r2, &c2);
while (c1! = r2)
{
 printf("error! column of first matrix not equal
 to row of second. \n");
 printf("enter rows & columns for first ma-
 trix:");
 scanf("%d, %d", &r1, &c1);
 printf("enter rows & columns for second
 matrix :");
 scanf("%d, %d", &r2, &c2);
}
printf("enter elements of matrix1 : \n");
for(i = 0; i < 1; ++i)
for(j = 0; j < c1; ++j)
{
   printf(enter elements a %d %d :", i+1, j+1);
   scanf("%d", & a[i] [j]);
}
printf("enter elements of matrix2 : \n");
for(i = 0; i < r2; ++i)
for(j = 0; j < c2; ++j)
{
   printf("Enter elements of b %d %d", i+1,
   j+1);
   scanf("%d", & b[i] [j]);
}
for(i = 0; i < r1; ++i)
for(j = 0; j < c2; ++j)
{
   m[i] [j] = 0;
}
for(i = 0; i < r1; ++i)
for(j = 0; j < c2; ++j)
for(k = 0; k < c1; ++k)
{
   m[i] [j] + = a[i] [k] * b[k] [j];
}
printf("\n output matrix : \n");
for(i = 0; i < r1; ++i)
for(j = 0; j < c2; ++j)
{
   printf("%d", m[i] [j]);
   if(j = c2 – 1)
   printf("\n \n");
}
}
```

Output :

```
Enter rows & columns of first matrix : 3
       4
Enter rows & columns for second matrix = 4
       5
Enter elements of matrix1 :
 Enter elements a11 : 2
 Enter elements a12 : 1
 Enter elements a13 : 3
 Enter elements a14 : 5
 Enter elements a21 : 4
 Enter elements a22 : 2
 Enter elements a23 : 5
 Enter elements a24 : 1
 Enter elements a31 : 4
 Enter elements a32 : 3
 Enter elements a33 : 5
 Enter elements a34 : 6
Enter elements of matrix2 :
 Enter elements b11 : 3
 Enter elements b12 : 1
 Enter elements b13 : 6
 Enter elements b14 : 4
 Enter elements b15 : 5
 Enter elements b21 : 7
 Enter elements b22 : 5
 Enter elements b23 : 3
 Enter elements b24 : 2
 Enter elements b25 : 6
 Enter elements b31 : 4
 Enter elements b32 : 5
 Enter elements b33 : 8
 Enter elements b34 : 3
 Enter elements b35 : 7
 Enter elements b41 : 5
 Enter elements b42 : 4
 Enter elements b43 : 5
 Enter elements b44 : 8
 Enter elements b45 : 3
Output matrix :
 50    42    64    59    52
 51    43    75    43    70
 83    68    103   85    91
```

Q8. Give the main advantage of storing data as a file. Describe various ways in which data files can be categorized in 'C'. Illustrate by using examples?

Ans. Files are used for permanent retention of large amounts of data. *File* is a collection of numbers, symbols and text placed on the disk. Files can be read and modified as per the user requirements. Thus, files allow us to store information permanently in the disk and to access. Further, it can be altered depending upon the needs. This process leads to the concept of *data files*.

There are main advantages of storing data as a file :

- Firstly, it is very essentia: to store data permanently. We cannot preserve the output of any program for future use without files. Note that all messages or values printed using output statements such as **printf(), putchar()** etc., are never available for future use. In that case, the only solution is to write the output in files.
- When there is a large amount of data generated as output by a program, storing that output in file will help in easy handling or analysis of the output. Thus, the user can see the whole output at any time, even after the complete execution of the program.
- If a program needs a lot of data to be feed as input, the user cannot keep on typing that again and again for repeated execution of the program. In such a situation, all input data can be once written in a file. Then that file can be easily used as the input file.
- The transfer of input data and / or output data from one computer to another can be easily done by using files.

Different categories of data files in C :

1. Stream oriented data files :

Such files are of 2 types :

(a) Text files :

(i) Data files comprise consecutive characters. These characters can be interpreted as individual data items or as components of strings or numbers.

(ii) A text file contains only textual information like alphabets, digits & special symbols.

(iii) In text files, the ASCII codes of alphabets, digits, special symbols are stored.

(iv) A good example of a text file is any C program, say program C is the only function that is available for storing number in the disk file is the fprintf( ) function.

(b) Binary files :

(i) Often called as unformatted data files, organizes data into blocks containing contiguous bytes of information. These blocks represent more complex data structures, such as arrays & structures.

(ii) A binary file is merely a collection of bytes.

(iii) This collection might be a compiled version of a C program say program1. exe or music data stored in a wave file or a picture stored in a graphic file.

(iv) The functions that are available for storing or receiving numbers in a binary file are the fwrite( ) & fread( ) function.

2. System oriented data files :

They are more closely related to the computer's operating system that are stream oriented data files. To perform the Input/Output form and to files, an extensive set of library functions are available in C.

Section – C

Q9(a). What are pre-processor directives? Explain various pre-processor directives?

Ans. Preprocessor Directives :

A C preprocessor is just a text substitution tool and they instruct compiler to do required preprocessing before actual compilation. Pre-processor directives are lines included in the code of programs preceded by a hash sign (#). It must be the first non-blank character, and for readability, a preprocessor directive should begin in first column. These lines are not the program statements, but the directives for the preprocessor. The preprocessor examines the code before actual compilation of code begins and resolves all these directives before any code is actually generated by regular statements.

These preprocessor directives extends only across a single line of code. As soon as a newline character