Module 3

Arrays and Strings

ARRAYS

Ques 1) Define one dimensional array. How it is represented in memory? What is its need?

Or

What is an array? Explain Different types of arrays with their initialization. Also discuss the merits and demerits of arrays.

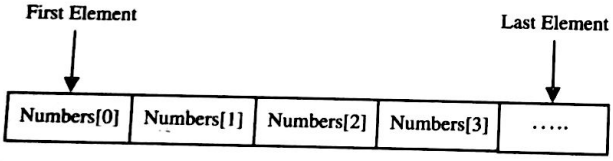
Or

Explain how to declare an array?

Ans: Array

Array means collection. In an array, similar types of data are stored and these data are stored in the contiguous locations. It is also known as 'composite data structure' because data are organised sequentially.

It is also known as homogeneous data structure because similar types of data are saved. So, one can say that an array is memories which store the similar type of data in the contiguous manner. **Figure below** shows the concept of an array:



In the **figure**, the first address space save the first element of the array and the highest address saves the last one.

Types of Arrays

1) **Single/One Dimensional Array:** Representation of array elements in a single row is known as one dimensional array. It is also known as linear array.

Representation of Array in Memory

Arrays in memory locations are located in contiguous memory cells. The size of these cells is dependent on the type of data in the memory. The pointer to the array points to individual memory locations. When the elements are entered in the array then the pointer is on the first location and then moves subsequently. Similarly when the elements are extracted the pointer moves downwards. The contiguous memory allocation leads to garbage collection generally or sometimes to even wastage of data. That's, why the dynamic memory allocation is used to. **Figure 3.1** shows the contiguous memory in array:

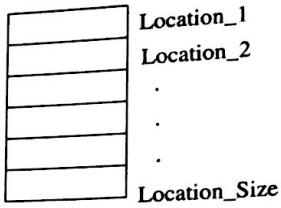


Figure 3.1: Contiguous Memory in Array

Needs of One Dimensional Array

When a list of similar items is to be processed, then one-dimensional array is used. **For example,** sorting and searching of an element.

Array Declaration/Array Notation

In a C program an array is declared before using it. Following is the syntax used to declare an array:

Syntax:

data_type array_name[size];

Where,

i) **data_type** indicates the type of data elements to be stored in the array.
ii) Name of array is **array_name**.
iii) Number of values to be stored in the array is indicated by the **size** and it is enclosed within the square bracket.

For example, let consider the following declarations:

```
int num[5];
char ch[5];
float m[10];
```

The size of an array can be computed by knowning the number of memory locations needed to individual data and number of data elements in the array(length). The size fo array int num[5]; is 10 because each integer needs two memroy locations to store it and length of the array is 5.

Thus, the size of the array is 2*5=10. Similarly, one can computer sized of character array, float arrays etc. 5 memory locations are needed for char ch[5] and 40 memory locations are needed for float m[10];

Initialisation of Single-Dimensional Array

Following are the statements in which shows the declaration and initialisation of array:

1) int students[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
2) int students[10] = {1, 2};
3) int students[10];

Explanation: In the first line code segment an integer type array is defined which has ten elements and all the elements are initialised (from value 1 to 10).

The second line code segment initialise only two values and all the remained elements do not have any value.

The last code segment no elements are defined and all are initialised to 0.

2) **Multi-Dimensional Array:** Array having more than one subscript variable is called Multi-Dimensional array. Multi-Dimensional Array is also called as Matrix.

Representation of Multi-Dimensional Array

An **array a[m][n]** is an m by n table having m rows and n columns containing m × n elements. The size of the array (total number of elements) is obtained by calculating m × n as shown in **figure 3.2:**

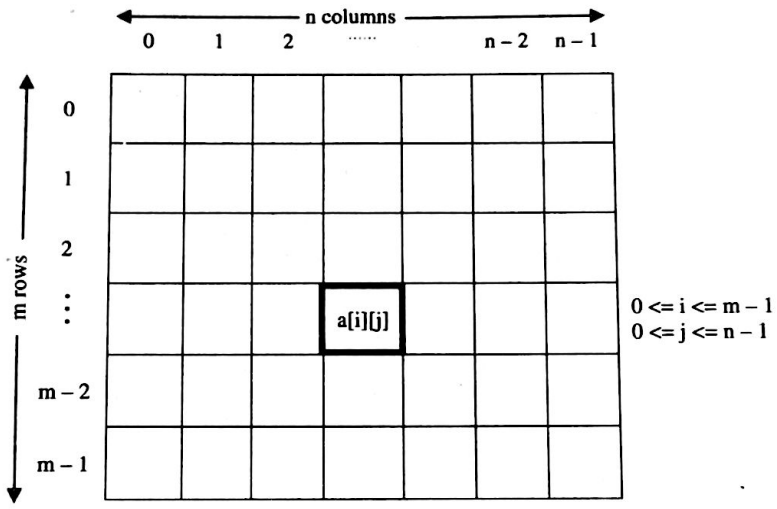


Figure 3.2

Here a[i][j] denotes the element in the i^{th} row and j^{th} column.

Syntax:

data-type <array_name> [row_subscript][column-subscript];

For example,

int a[3][3];

Initialisation of Multi-Dimensional Array

For initializing multi-dimensional array, one can need to assign values to each element of an array using the below syntax:

```
double x[3][2][4]={
{{-0.1, 0.22, 0.3, 4.3}, {2.3, 4.7, -0.9, 2}},
 {{0.9, 3.6, 4.5, 4}, {1.2, 2.4, 0.22, -1}},
 {{8.2, 3.12, 34.2, 0.1}, {2.1, 3.2, 4.3, -2.0}}
};
```

Merits of Arrays

1) All sorts of lists can be presented by using array.
2) Various other data structures such as stacks, queues, heaps, etc., are implemented by using array.

Demerits of Arrays

1) It is necessary to know the number of elements in an array.
2) Allocated memory cannot be extended or reduced.

Ques 2) What are two dimensional arrays? Write the syntax of two-dimensional array with example. Also discuss the initialisation of two-dimensional array.

Or

Explain how to declare two dimensional arrays with example.

Ans: Two-Dimensional Arrays

Sometimes programmer needs to save the data in the tabular form. To store the data in this form C has the concept of two dimensional arrays. It is also known as matrix.

Multi-dimensional arrays' simplest form is two-dimensional array. The two dimensional array is represented in row and column form and array elements are identified by the two subscripts.

For example, an array a[m][n] is an m by n table having m rows and n columns containing m × n elements. The size of the array (total number of elements) is obtained by calculating m × n.

Declaration of Two-Dimensional Arrays
Following are the statements which show the declaration of array:

Syntax:
type variable_name[number of rows][number of columns];

For example,
int a[5][5];

In the above example an integer array 'a' is defined which has the size of 5 by 5. So, the elements of array are:
a[0][0], a[0][1], ….. , a[4][4].

Initialisation of Two-Dimensional Array
To initialise the array following syntax is used:

Syntax:
type array_name[row_size][column_size] = {array_element1, array_element2…….. array_element_n}

For example, let us take a matrix A which is having 3 rows and 4 columns.

		Columns			
		1	2	3	4
Row	1	A[1, 1]	A[1, 2]	A[1, 3]	A[1, 4]
	2	A[2, 1]	A[2, 2]	A[2, 3]	A[2, 4]
	3	A[3, 1]	A[3, 2]	A[3, 3]	A[3, 4]

Two Dimensional 3 × 4 Array A

Ques 3) Explain the processing of array with suitable example. Also write a program to explain the processing of array.

Ans: Processing of Array
C language uses single index to access individual elements in a one-dimensional array. **For example,** given the marks in an array the first element is accessed as:

marks [0]

To process all the elements in marks, a loop similar to the following code is used:

```
for (i = 0; i < 10; i++)
{
    process(marks [i]);
}
```

Where, i is an integer variable.

The data represented in an array is actually stored in the contiguous memory cells of the machine. Because computer memory is linear, a one-dimensional array can be mapped on to the memory cells in a rather straight forward manner.

Storage for element marks[i + 1] will be adjacent to storage for element marks [i] for i = 1, 2, ... 10. To find the actual address of an element one simply needs to subtract one from the position of the desired entry and then add the result to the address of the first cell in the sequence.

The array's name is a symbolic reference for the address to the first byte of the block of memory allocated for the array. The address of the first byte for the array is also known as **base address** of the array.

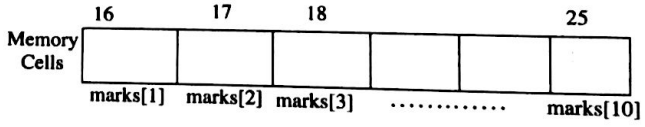
Whenever programmers use the array's name, we actually refer to the first byte of the array. The index represents an offset from the beginning of the array to the element being referenced. With these two facts of information, the C compiler can calculate the address of any element in the array using the following simple formula:

Element address = base address + (size of (element) * index)

Or

Address (marks[i]) = base address + (size of (element) * i)

For example, consider an array of marks of 10 elements. The address of marks[4]. If the first cell in the sequence marks[1], marks[2], ... marks[10], was at address 16, then A[4] would be located at 16 + (4 − 1) = 19, as shown in **figure below**. We assume that the size of each element stored is one unit.



Here is the section of code that places data into an array to displaying output:

```
for (i=0; i<=29; i++)
{
    printf ("\nEnter marks");
    scanf ("%d", &marks[i]);
}
```

The **for** loop causes the process of asking for and receiving a student's marks from the user to be repeated 30 times. The first time through the loop, **i** has a value 0, so the **scanf()** statement will cause the value typed to be stored in the array element **marks[0]**, the first element of the array.

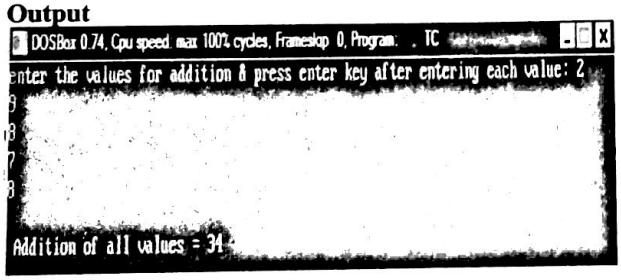
This process will be repeated until **i** become 29. This is last time through the loop, because there is no array element like **marks[30]**.

In the **scanf()** statement, we have used the "address of" operator (&) on the element **marks[i]** of the array. In so doing, we are passing the address of this particular array element to the **scanf()** function, rather than its value; which is what **scanf()** requires.

Program: Illustrating Processing of Array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i[4]; int j; int Addition=0;
    clrscr();
    printf("enter the values for addition & press enter key
    after entering each value: ");
    for(j=0; j<=4; j++)
        {
        scanf("%d", &i[j]);
        }

    for(j=0; j<=4; j++)
        {
        Addition = Addition+i[j];
        }

    printf("\n Addition of all values = %d", Addition);
    getch();
}
```

Output



Ques 4) What are the main applications of arrays?

Ans: Applications of Arrays

Arrays are most useful when they have a large number of elements: that is, in cases where it would be completely impractical to have a different name for every storage space in the memory.

1) **Contiguous Memory Allocation:** Array is used to allocate the contiguous memory allocation of other data structures. The contiguous memory reduces the overheads of keeping pointers.

2) **Indexed Storage:** The storage spaces in arrays have indices. These numbers can often be related to variables in a program and so there is a logical connection to be made between an array and a program. Hence sequential way of accessing elements is enhanced.

3) **Grouping of Similar Variables:** In C, arrays can be initialized very easily indeed. It is far easier to initialize an array than it is to initialize twenty or so variables. So it reduces multiple initialization of same variable of same data type.

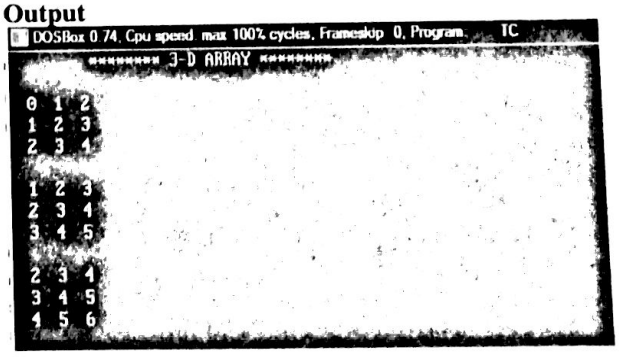
4) **Mathematical Operations on Matrices:** The matrices can be represented by using arrays by exploiting the continuous memory property of array. The array representation of matrices can be employed on functions of addition, transpose and etc.

Ques 5) Write a program to display three dimensional arrays?

Ans: Program Illustrating Three-Dimensional Array

```
# include<stdio.h>
# include<conio.h>
void main()
{
int array_3d[3][3][3];
int a,b,c;
clrscr();
printf("\t ******** 3-D ARRAY ********\n");
for (a=0;a<3;a++)
   for(b=0;b<3;b++)
      for(c=0;c<3;c++)
         array_3d[a][b][c]=a+b+c;
for(a=0;a<3;a++)
 {
 printf("\n");
      for (b=0;b<3;b++)
       {
          for (c=0;c<3;c++)
          printf("%3d", array_3d[a][b][c]);
          printf("\n");
        }
   }
   getch();
}
```

Output



Explanation: The three-dimensional array called array_3d in the code is initialized. The first for loops are used for adding the values of a, b, and c. Here, initially a and b are zero and 'c' varies from 0 to 2. Hence, the addition of a, b and c will be 0 1 2. This will be printed in the first row. The second output row in which a=0, b=1 and c varies from 0 to 2. Thus, the output of second row will be 2 3. In this way the values of a, b and c are changed and the total 27 iterations are carried out.

STRINGS

Ques 6) What are strings? Explain with examples.
Or
What is a string in C language? Explain how to declare and initialize the string in C language.

Ans: Strings in C Language
A string is a sequence of symbols that are chosen from a set of alphabet. A string is, essentially, a sequence of characters.

Strings in C are represented by arrays of characters. The end of the string is marked with a special character, the null character, which is simply the character with the value 0. **For example,** we can declare and define an array of characters, and initialize it with a string constant:

```
char string[] = "Hello, world!";
```

String are used to suggest strings in which the stored data does not (necessarily) represent text.

A variable declared to have a string data type usually causes storage to be allocated in memory that is capable of holding some predetermined number of symbols.

When a string appears literally in source code, it is known as a **string literal** and has a representation that denotes it as such.

For example,

```
char unit;
unit = 'F';    /* for Fahrenheit */
.....
unit = 'C';    /* for Centigrade */
```

Most codes and ID's require more than one character. In mailing addresses, two characters are used to identify a state and five characters are used to identify a zip code. They could be stored in character array as follows:

```
char state[2], zip[5];
```

Numbers such as zip codes, which are used for identification purposes and are not used in arithmetic, should be stored as arrays of type char, rather than as integers.

Declaration of Strings
In C, a character array is declared using the basic data type **char** and the derived data type array.

Syntax:

```
char StringName[array_size];
```

For example, char chararray[n];

The data type here is char, the name of the array is chararray, and n is the number of elements.

For example, consider the following declaration and allocation of storage for a character array that can hold 10 characters.

```
char chararray[10];
chararray[0] = 'a';
chararray[1] = 'x';
chararray[2] = 'c';
chararray[3] = 'b';
chararray[4] = 'l';
chararray[5] = 'k';
chararray[6] = 'm';
chararray[7] = 'd';
chararray[8] = 'g';
chararray[9] = 'f';
```

The preceding characters are stored as follows:

a	x	c	b	l	k	m	d	g	f

Here in above example 10 characters are stored in this array **chararray**. As with any array these characters can only be accessed individually.

Initialization of Strings
Initialization of character data is similar to that of other types of arrays, since most character data is stored in arrays of type **char**. An array can be initialized by explicit assignment of values to its elements.

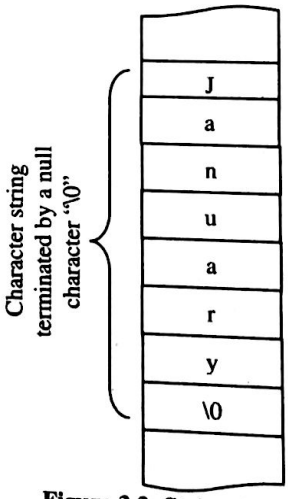


Figure 3.3: String Stored in a String Variable Month

Example: An array can be initialized in the array declaration as shown in the following:

```
char mystate[2] = {'C', 'A'};
char myzip[5] = {'9', '2', '3', '1', '2'}; //here '9', '2'...
```

are treated as characters not integers.

Example: Initialization of a string must have the following form, (this is similar to initializing a one-dimensional array):

```
char month1[] = { 'J', 'a', 'n', 'u', 'a', 'r', 'y', '\0' };
```

Then the string month is initialized to January. This is perfectly valid, but C offers a special way to initialize strings. The above string can be initialized as:

```
char month1[] = "January";
```

The characters of the string are enclosed within a pair of double quotes. The compiler takes care of storing the ASCII codes of the characters of the string in memory and also stores the null terminator in the end. It is a one-dimensional array. Each character occupies a byte. A null character ('\0') that has the ASCII value 0 terminates the string. **Figure 3.9** shows the storage of the string January in the memory. Recall that \0 specifies a single character whose ASCII value is zero. Number of literals must match the number of elements in the array. If too few literals are specified, the rest of the array is filled with blank characters. Also, a number being stored in a char array must be specified as individual digits.

Ques 7) Explain any four string functions with suitable example in C language.

Or

Explain string handling functions with its syntax. Also write a program illustrating string handling functions.

Ans: String Handling Functions

1) **strlen():** This function is commonly used to determine the length of a string. This is a single argument function. The argument may be a string variable or a string literal. The function strlen() returns an integer that is the count of the number of characters in the string including embedded blanks and excluding the null terminator.

   Syntax:

   strlen(String);

2) **strcpy():** The function strcpy() is used to copy one string to another that is, it copies of one variable of string type to another variable of string type.

   Syntax:

   strcpy(String1, String2);

3) **strcat():** The process of combining of two strings together is technically termed as **concatenation.** It is done by the library function called **strcat()**, which concatenates strings. It does not concatenate two strings together and returns a third, but it appends one string onto the end of another.

   Syntax:

   strcat(String1, String2);

4) **strupr():** This function takes string variable as its single argument and returns a pointer to converted string. It converts the lowercase alphabet letters of the string to uppercase letters.

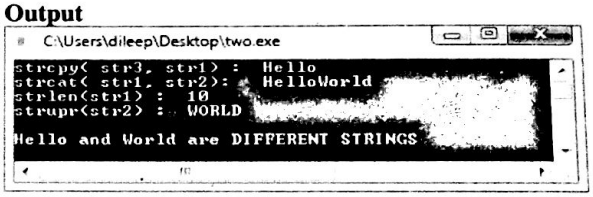
   Syntax:

   strupr(st1);

Program: Illustrating String Handling Functions

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
int main (){
  char str1[12] = "Hello";
  char str2[12] = "World";
  char str3[12];
  int len ;
  strcpy(str3, str1);  /* copy str1 into str3 */
  printf("strcpy( str3, str1) :  %s\n", str3 );
  strcat( str1, str2); /* concatenates str1 and str2 */
  printf("strcat( str1, str2):  %s\n", str1 );
  /* total length of str1 after concatenation */
  len = strlen(str1);
  printf("strlen(str1) :  %d\n", len );
  printf("strupr(str2) :  %s\n", strupr(str2));
  if(strcmp(str1, str2) == 0)
     printf("\nHello and World are EQUAL
     STRINGS\n",str1,str2);
  else
     printf("\nHello and World are DIFFERENT
STRINGS\n",str1,str2);
  getch(); }
```

Output

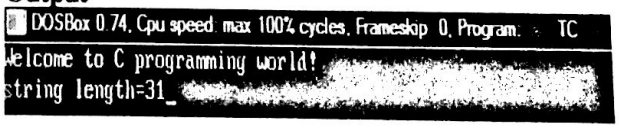


Ques 8) Write a program to print the length of the strings without using strlen() function.

Ans: Program for Printing String Length Without Using strlen().

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char a[20];
    int i;
    clrscr();
    gets(a);
    i=0;
    while(a[i]!='\0')
    i++;          //counts no of chars till encountering null
    char
    printf("string length=%d",i);
    getch(); }
```

Output



Ques 9) What is the use of strcmp() function in string handling?

Ans: strcmp() Function

This C library function compares the string. strcmp() compares two string and returns value 0, if the two strings are equal.

Syntax:

```
int k = strcmp(strng1, strng2);
```

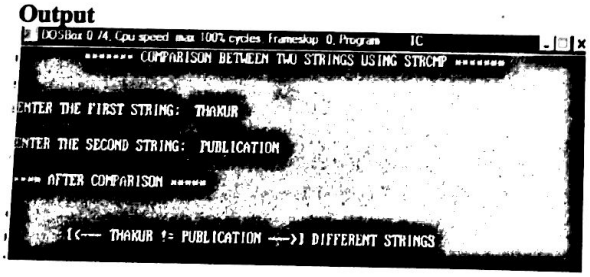
In this case, the character strings strng1 and strng2 are compared character-by-character. When unequal characters are found the order of the two strings is determined. Based on the order of the two strings the comparison generates the following results:

1) k > 0 positive number i.e., (strng1 after strng2)
2) k = 0 positive number i.e., (strng1 identical to strng2)
3) k < 0 negative number i.e., (strng1 before strng2)

Program: String Comparison Using strcmp() Function.

```
#include<string.h>
#include<conio.h>
#include<string.h>
void main()
{
    char st1[25],st2[30];
    clrscr();
    printf("\t ******* COMPARISON BETWEEN TWO STRINGS USING STRCMP *******\n\n");
    printf("\nENTER THE FIRST STRING: ");
    gets(st1);
    printf("\nENTER THE SECOND STRING: ");
    gets(st2);
    printf("\n**** AFTER COMPARISON *****\n");
    if(strcmp(st1,st2) == 0)
    printf("\n\n\t[<--- %s == %s --->] EQUAL STRINGS\n",st1,st2);
    else
    printf("\n\n\t[<--- %s != %s --->] DIFFERENT STRINGS\n",st1,st2);
    getch();
}
```

Output



Ques 10) Discuss about the strstr() function with suitable example.

Ans: strstr() Function

The function strstr() searches one string for the occurrence of another. It accepts two strings as parameters and searches the first string for an occurrence of the second. The function returns the memory address of the character in the first string, starting at which the first occurrence of the second string is found.

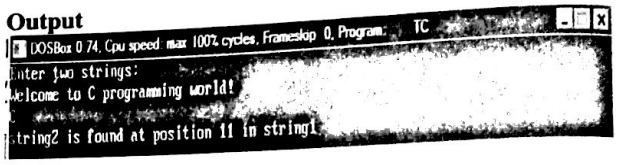
Syntax:

```
strstr(String1, String2);
```

Program: Searching for a substring within String

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

void main() {
    char str1[30], str2[30];
    char *found;
    clrscr();
    printf("Enter two strings:\n");
    gets(str1);
    gets(str2);
    /* Search for str2 in str1 */
    found=strstr(str1,str2);
    printf("string2 is found at position %d in string1\n",found-str1); }
```

Output



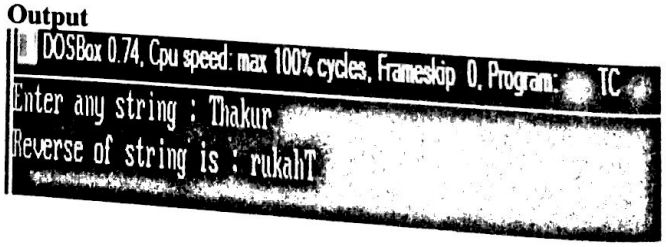
Explanation: Here, the first string passed to strstr is, **Welcome to C programming world!**. The second string is **C**. The function searches the first string for the given second string. Though there are one instances of **C** in the first string, so it is considered, and a pointer to the beginning of the match is returned.

Ques 11) Write a progam to reverse the strings without using strrev() function.

Program: Reversing a string in c without using string function (strrev).

```
#include<stdio.h>
#include<conio.h>
int main() {
    char str[50];
    char rev[50];
    clrscr();
    int i=-1,j=0;
    printf("Enter any string : ");
    scanf("%s",str);
    while(str[++i]!='\0');
    while(i>=0)
    rev[j++] = str[--i];
    rev[j]='\0';
    printf("Reverse of string is : %s",rev);
    return 0; }
```

Output



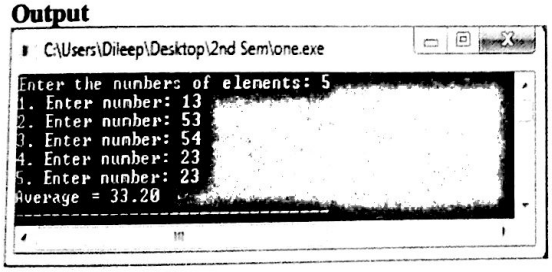
EXAMPLE PROGRAMS

Ques 12) Write a program to calculate average using arrays.

Ans: Program to Calculate Average

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int n, i;
  float num[100], sum = 0.0, average;

  printf("Enter the numbers of elements: ");
  scanf("%d", &n);

  while (n > 100 || n <= 0)
  {
    printf("Error! number should in range of (1 to 100).\n");
    printf("Enter the number again: ");
    scanf("%d", &n);
  }

  for(i = 0; i < n; ++i)
  {
    printf("%d. Enter number: ", i+1);
    scanf("%f", &num[i]);
    sum += num[i];
  }

  average = sum / n;
  printf("Average = %.2f", average);

  return 0;
}
```

Output



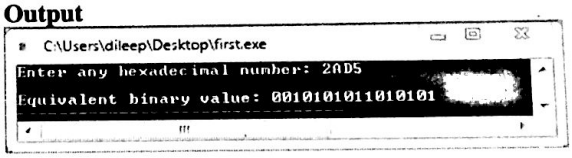
Explanation: This program takes the number of elements in the array and stores in the variable **n.** Then, the **for** loop gets all the elements from the user and stores the sum of the entered numbers in **sum**. Finally, the average is calculated by dividing **sum** by the number of elements **n**.

Ques 13) Write a program to convert a given four digit hexadecimal number into binary.

Ans: C Program for Hexadecimal to Binary Conversion.

```
#include<stdio.h>
#define MAX 1000
int main()
{
  char binaryNumber[MAX],hexaDecimal[MAX];
  long int i=0;
  printf("Enter any hexadecimal number: ");
  scanf("%s",hexaDecimal);
  printf("\nEquivalent binary value: ");
  while(hexaDecimal[i]){
    switch(hexaDecimal[i]){
      case '0': printf("0000"); break;
      case '1': printf("0001"); break;
      case '2': printf("0010"); break;
      case '3': printf("0011"); break;
      case '4': printf("0100"); break;
      case '5': printf("0101"); break;
      case '6': printf("0110"); break;
      case '7': printf("0111"); break;
      case '8': printf("1000"); break;
      case '9': printf("1001"); break;
      case 'A': printf("1010"); break;
      case 'B': printf("1011"); break;
      case 'C': printf("1100"); break;
      case 'D': printf("1101"); break;
      case 'E': printf("1110"); break;
      case 'F': printf("1111"); break;
      case 'a': printf("1010"); break;
      case 'b': printf("1011"); break;
      case 'c': printf("1100"); break;
      case 'd': printf("1101"); break;
      case 'e': printf("1110"); break;
      case 'f': printf("1111"); break;
       default:  printf("\nInvalid hexadecimal digit %c
       ",hexaDecimal[i]); return 0;  }
       i++;  }
 return 0;}
```

Output

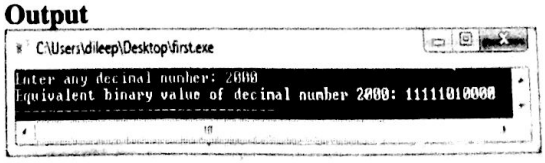


Ques 14) Write a program to convert a given four digit decimal number into binary.

Ans: C Program for Decimal to Binary Conversion

```
#include<stdio.h>
int main()
{  long int decimalNumber,remainder,quotient;
   int binaryNumber[100],i=1,j;
   printf("Enter any decimal number: ");
   scanf("%ld",&decimalNumber);
   quotient = decimalNumber;
   while(quotient!=0)
   {    binaryNumber[i++]= quotient % 2;
        quotient = quotient / 2;
   }printf("Equivalent binary value of decimal number   %d:
",decimalNumber);
   for(j = i -1 ;j> 0;j--)
       printf("%d",binaryNumber[j]);
   return 0;}
```

Output



Ques 15) Write a program to add two matrices of dimension 3*3 and store the result in another matrix.

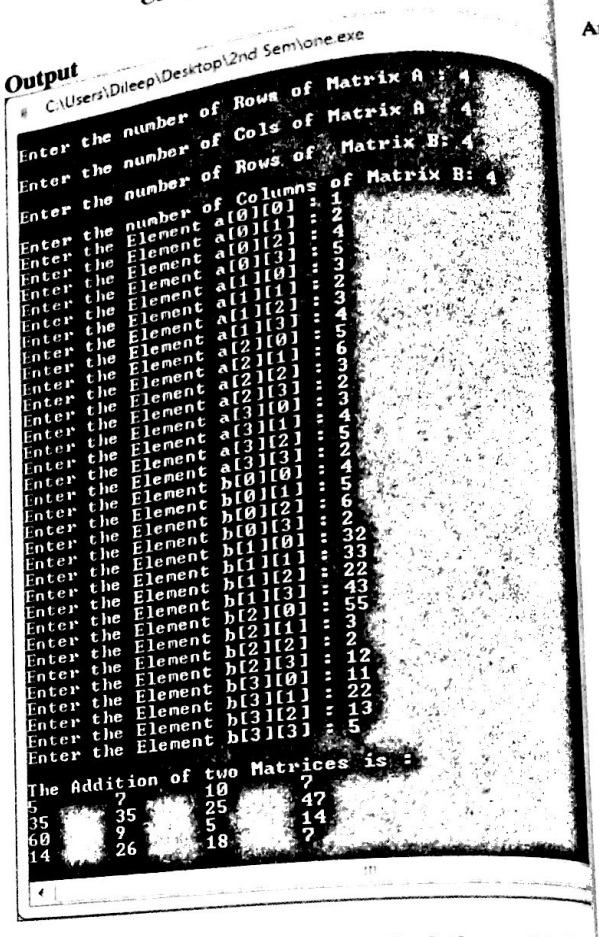
Or

Write a program to add the two matrixes A[4 × 4] and B[4 × 4] of positive integers. The result should be stored in matrix R.

Ans: C Program for Addition of Two Matrices

```
#include<stdio.h>
#include<stdio.h>
int main() {
  int i, j, A[10][10], B[10][10], R[10][10];
  int row1, col1, row2, col2;
  printf("\nEnter the number of Rows of Matrix A : ");
  scanf("%d", &row1);
  printf("\nEnter the number of Cols of Matrix A : ");
  scanf("%d", &col1);

  printf("\nEnter the number of Rows of  Matrix B: ");
  scanf("%d", &row2);
  printf("\nEnter the number of Columns of Matrix B: ");
  scanf("%d", &col2);

  /* Before accepting the Elements Check if no of
   rows and columns of both matrices is equal */
  if (row1 != row2 || col1 != col2) {
    printf("\nOrder of two matrices is not same ");
    exit(0);
  }

  //Accept the Elements in Matrix A
  for (i = 0; i < row1; i++) {
    for (j = 0; j < col1; j++) {
      printf("Enter the Element a[%d][%d] : ", i, j);
      scanf("%d", &A[i][j]);
    }}

  //Accept the Elements in Matrix B
  for (i = 0; i < row2; i++)
    for (j = 0; j < col2; j++) {
      printf("Enter the Element b[%d][%d] : ", i, j);
      scanf("%d", &B[i][j]); }

  //Addition of two matrices
  for (i = 0; i < row1; i++)
    for (j = 0; j < col1; j++)
    {
      R[i][j] = A[i][j] + B[i][j];
  }

  //Print out the Resultant Matrix R
  printf("\nThe Addition of two Matrices is : \n");
  for (i = 0; i < row1; i++) {
    for (j = 0; j < col1; j++) {
      printf("%d\t", R[i][j]); }
    printf("\n");}
getch();
 return (0);}
```

Output



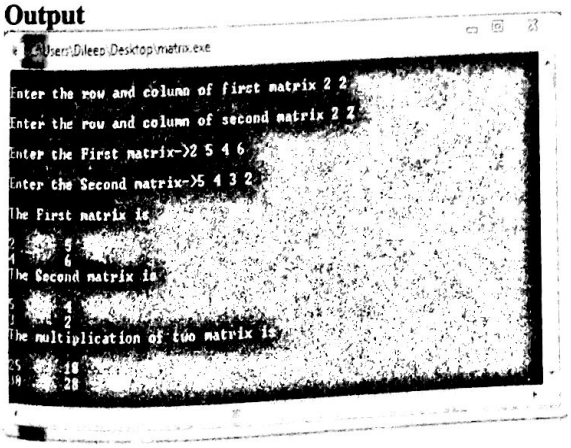
Ques 16) Write a C program to find the multiplicati of two matrices.

Ans: C Program for Matrix Multiplication

```
#include<stdio.h>
#include<conio.h>
int main()
{   int a[5][5],b[5][5],c[5][5], i, j, k, sum=0,m,n,o,p;
    printf("\nEnter the row and column of first matrix");
    scanf("%d %d", &m, &n);
    printf("\nEnter the row and column of second matrix");
    scanf("%d %d",&o,&p);
    if(n!=o)
    {
        printf("Matrix mutiplication is not possible");
        printf("\nColumn of first matrix must be same
        row of second matrix");
    }
    else
    {
        printf("\nEnter the First matrix->");
        for(i=0;i<m;i++)
        for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);
        printf("\nEnter the Second matrix->");
        for(i=0;i<o;i++)
        for(j=0;j<p;j++)
        scanf("%d",&b[i][j]);
        printf("\nThe First matrix is\n");
```

```
        for(i=0;i<m;i++)
        {
            printf("\n");
            for(j=0;j<n;j++)
            {
                printf("%d\t",a[i][j]);
            }
            }
        printf("\nThe Second matrix is\n");
        for(i=0;i<o;i++)
        {
            printf("\n");
            for(j=0;j<p;j++)
            {
                printf("%d\t",b[i][j]);
            }
            }
        for(i=0;i<m;i++)
        for(j=0;j<p;j++)
        c[i][j]=0;
        for(i=0;i<m;i++)
        { //row of first matrix
            for(j=0;j<p;j++)
            { //column of second matrix
                sum=0;
                for(k=0;k<n;k++)
                sum=sum+a[i][k]*b[k][j];
                c[i][j]=sum;
            }
            }
            }

    printf("\nThe multiplication of two matrix is\n");
    for(i=0;i<m;i++)
    {
        printf("\n");
        for(j=0;j<p;j++)
        {
            printf("%d\t",c[i][j]);
        }}
    getch();
    return 0;
}
```

Output



Ques 17) Write a C program using functions with proper parameters which reads values of diagonal elements of a square matrix and then copies the value of first diagonal element into first row as well as first column. Value of second element is copied into all remaining elements of 2nd row as well as 2nd column and so on. Display the complete matrix as output. For example, if a square matrix of 4*4 matrix has diagonal elements as 4, 20, 9, 7 then the output matrix should be:

4	4	4	4
4	**20**	20	20
4	20	**9**	9
4	20	9	**7**

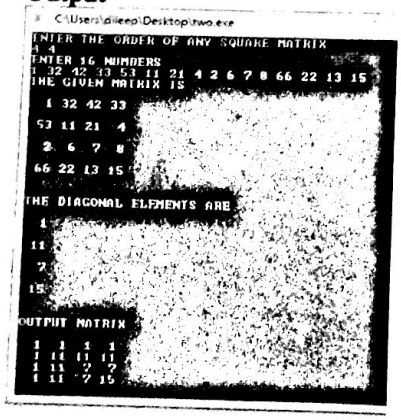
Values shown in bold will be read by the program and rest if the values should be generated by the program.

Ans: Program Printing Diagonal Elements

```
#include <stdio.h>
main()
{
int a[5][5],i,j,m,n, temp1, temp2, temp3,temp4;
printf("ENTER THE ORDER OF ANY SQUARE
MATRIX\n");
scanf("%d%d",&m,&n);
printf("ENTER %d NUMBERS\n",m*n);
for (i=0;i<m;i++)
  for (j=0;j<n;j++)
   scanf("%d",&a[i][j]);

printf("THE GIVEN MATRIX IS\n\n");
for (i=0;i<m;i++)
{
  for (j=0;j<n;j++)
   printf("%3d",a[i][j]);
  printf("\n\n");
}
printf("\nTHE DIAGONAL ELEMENTS ARE\n\n");
for(i=0;i<m;i++)
{
   for(j=0;j<n;j++)
   { if (i==j)
     printf("%3d",a[i][j]);
     if (i==0 && j==0)
     {
          temp1 = a[i][j];
     }
     else if(i==1 && j==1)
     {
          temp2 = a[i][j];
     }
     else if(i==2 && j==2)
     {
          temp3=a[i][j];
     }
     else
          temp4=a[i][j];

   }
```

```
    printf("\n\n");
  }
  printf("\nOUTPUT MATRIX\n\n");
  for(i=0;i<m;i++)
  {
    for(j=0;j<n;j++)
    {
      if(i==0 && j>=i)
      {
          a[i][j]= temp1;
          printf("%3d",temp1);
      }
      else if(j==0 && j<i)
        {
          a[i][j]= temp1;
          printf("%3d",temp1);
      }
      else if(i==1 && j>i)
      {
          a[i][j]= temp2;
          printf("%3d",temp2);
      }
      else if(j==1 && j<=i)
        {
          a[i][j]= temp2;
          printf("%3d",temp2);
      }
      else if(i==2 && j>i)
      {
          a[i][j]= temp3;
          printf("%3d",temp3);
      }
      else if(j==2 && j<=i)
      {
          a[i][j]= temp3;
          printf("%3d",temp3);
      }
      else
          printf("%3d",temp4);
    }
    printf("\n");
  }
  getch();
  }
```

Output

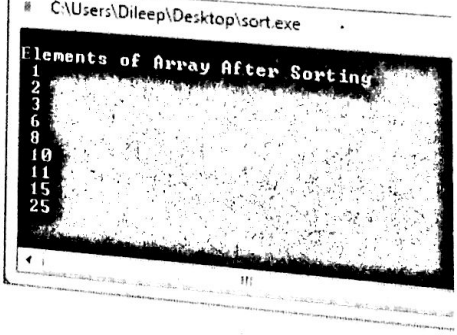


Ques 18) Take two single dimensional sorted arrays of integers, A and B of different size. Write a program in C language to merge these sorted arrays into a third array C such that all the elements are sorted and no element should be repeated.

Ans: C Program for Merging and Sorting the Elements of two Sorted Arrays

```
#include<stdio.h>
#include<stdio.h>
int A[5] = {3, 6, 8, 10, 11};
int B[5] = {1, 2, 6, 15, 25};
int C[10] = {0};

void merge()
{   int al_size = 0, a2_size = 0, res_size = 0;
    while(al_size<5 && a2_size<5)
    {
        if(A[al_size] < B[a2_size])
        {   C[res_size] = A[al_size];
            al_size += 1;
            res_size += 1; }
        else if(A[al_size] > B[a2_size])
        {   C[res_size] = B[a2_size];
            a2_size += 1;
            res_size += 1; }
        else if(A[al_size] == B[a2_size]) {    a2_size += 1; }
        else {
            C[res_size] = B[a2_size];
            a2_size += 1;
            res_size += 1;
            C[res_size] = A[al_size];
            al_size += 1;
            res_size += 1; }}

    while(al_size<5) {   C[res_size] = A[al_size];
        al_size += 1;
        res_size += 1; }

    while(a2_size<5) {
        C[res_size] = B[a2_size];
        a2_size +=1;
        res_size +=1;  }}

int main(void) {
    int i;
    merge();
    printf("\nElements of Array After Sorting");
    for(i = 0; i < 9; i++)
    printf("\n %d", C[i]);
    getch();
    return 0; }
```

Output

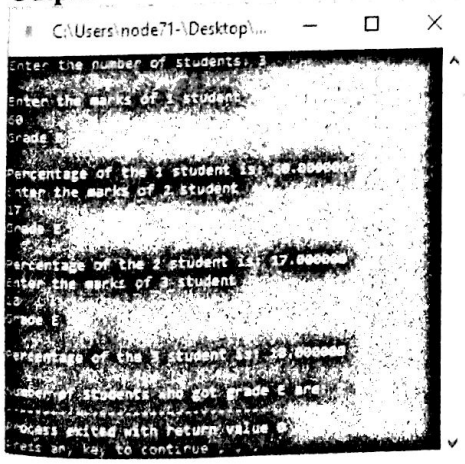


Ques 19) Create an array to store the marks of 500 students in a subject. Maximum mark in the subject is 100. Write a program in C language to convert the marks into grades as follows: marks <20 is 'E' grade; 20>=marks <40 is 'D' grade; 40>=marks <60 is 'C' grade; 60>= marks <80 is 'B' grade; 80>= marks <=100 is A grade. Calculate the percentage of marks of each student. Also give the total number of students who have secured 'E' grades.

Ans: C Program to Convert Marks into Grades

```
#include<stdio.h>
#include<conio.h>
int main() {
    int marks[500],n, count=0;
    int i;
    float per;
    printf("Enter the number of students: ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {   printf("\nEnter the marks of %d student\n", i+1);
        scanf("%d",&marks[i]);
        if(marks[i] >= 80&&marks[i] <= 100)
        {   printf("Grade A");
        }
        else if(marks[i] >= 60&&marks[i] < 80)
        {   printf("Grade B");
        }
        else if(marks[i] >= 40&&marks[i] < 60)
        {   printf("Grade C");
        }
        else if(marks[i] >= 20&&marks[i] < 40)
        {   printf("Grade D");
        }
        else if(marks[i] < 20)
        {   printf("Grade E"); }
        per=marks[i]*100/100;
        printf("\n\nPercentage of the %d student is: %f",
        i+1, per);

        if(marks[i]<20) {
            count++; }}

    printf("\n\nNumber of Students who got grade E are:
    %d", count);
    return 0;
}
```

Output

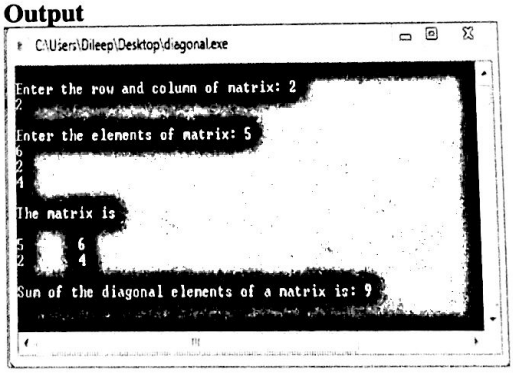


Ques 20) Write a program to compute the sum of diagonal elements of a square matrix.

Ans: Program to Compute the Sum of Diagonal Elements of a Square Matrix

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a[10][10],i,j,sum=0,m,n;
    printf("\nEnter the row and column of matrix: ");
    scanf("%d %d",&m,&n);
    printf("\nEnter the elements of matrix: ");
    for(i=0;i<m;i++)
      for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);
        printf("\nThe matrix is\n");

    for(i=0;i<m;i++)
    {   printf("\n");
        for(j=0;j<m;j++)
        {  printf("%d\t",a[i][j]);
        } }
    for(i=0;i<m;i++)
    {   for(j=0;j<n;j++)
      {
        if(i==j)
            sum=sum+a[i][j];
      }}
printf("\n\nSum of the diagonal elements of a matrix is:
%d",sum);
getch();
return 0;
}
```

Output



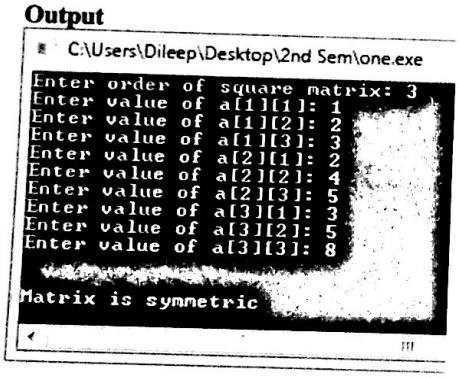
Ques 21) Write a C program to check whether a given square matrix is symmetric or not.

Ans: C Program to Check Whether a Given Square Matrix is Symmetric or Not

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a[10][10],i,j,m;
    printf("Enter order of square matrix: ");
    scanf("%d",&m);
    for(i=1;i<=m;i++)
```

```
    {
        for(j=1;j<=m;j++)
        {
            printf("Enter value of a[%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);
        }
    }

    for(i=1;i<=m;i++)
    {
        for(j=1;j<=m;j++)
        {
            if(a[i][j]!=a[j][i])
            {
                printf("\n\nMatrix is not symmetric");
                getch();
                exit(0);
            }
        }
    }

    printf("\n\nMatrix is symmetric");
    getch();
    return 0;
}
```

Output

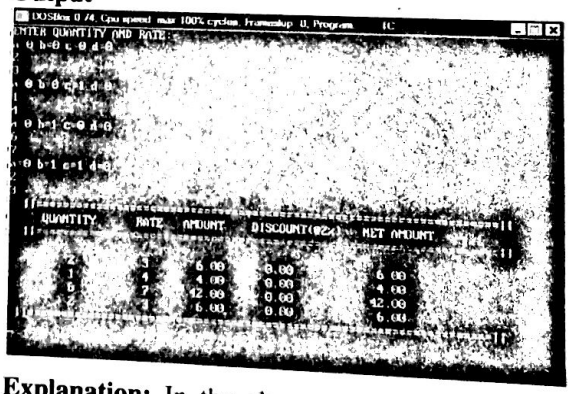


Ques 22) Read the quantity and rate of certain items using multidimensional array. Calculate total cost by multiplying quantity and rate and offer 2% discount on it and display the net amount.

Ans: Program to Display Net Amount

```
# include<stdio.h>
# include<conio.h>
main ()
{
long m[2] [2] [2] [2];
int a,b,c,d;
printf("\t ******* QUANTITY & RATE USING MULTI-D
ARRAY ********\n");
clrscr();
printf("ENTER QUANTITY AND RATE:- \n");
for(a=0;a<2;a++)
  for(b=0;b<2;b++)
    for(c=0;c<2;c++)
     for(d=0;d<1;d++)
     {
       if (a==0)
       {
        printf("a=%d b=%d c=%d d=%d\n",a,b,c,d);
        scanf("%ld %ld",&m[a][b][c][d],&m[a][b][c][d+1]);
       }
       else
       {
        m[a][b][c][d]=m[a-1][b][c][d]*m[a-1][b][c][d+1];
        m[a][b][c][d+1]=m[a-1][b][c][d]*m[a-
1][b][c][d+1]*2/100;
       }

}
printf("
||==========================================
==========||\n");
printf("          QUANTITY          RATE       AMOUNT
DISCOUNT(@2%)  NET AMOUNT\n");
printf("
||==========================================
==========||\n");
for(a=0;a<1;a++)
  for(b=0;b<2;b++)
    for(c=0;c<2;c++)
      for(d=0;d<1;d++)
       {
        printf("\n%10ld
%10ld",m[a][b][c][d],m[a][b][c][d+1]);
        printf("%8ld.00                    %6ld.00
%12ld.00",m[a+1][b][c][d],m[a+1][b][c][d+1],m[a+1][b][c][
d]-m[a+1][b][c][d+1]);
       }
printf("\n
||==========================================
==========||\n");
getch();
}
```

Output



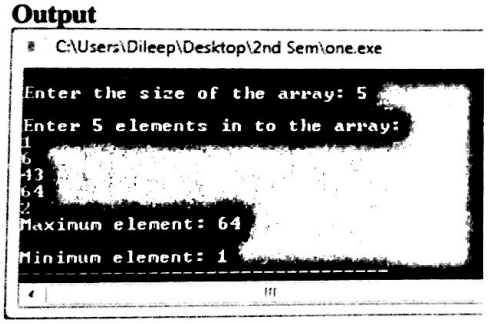
Explanation: In the above example the four-dimension array m[2] [2] [2] [2] is declared. The first four for loops are used to read the quantity and rate. The values of variables 'a' and 'd' for four times remains zero whereas values of 'b' and 'c' changes to 1, (0,0), 2 (0,1), 3 (1,0) and 4 (1,1). This happens during execution of for loops. The if statement checks as to whether the value of a==0 or not. As long as its value is zero reading operation is performed. When it is greater than zero, rate and quantity are multiplied for obtaining the amount. Also discount is calculated. Amount and discount are stored in the array m[2] [2] [2] [2]. Net amount is printed after subtracting discount from gross amount.

Ques 23) Write a program to find the minimum and maximum element of the array.

Ans: Program to Find the Minimum and Maximum Element of the Array

```
#include<stdio.h>
int main(){
 int a[50],size,i,big,small;

 printf("\nEnter the size of the array: ");
 scanf("%d",&size);
 printf("\nEnter %d elements in to the array: \n", size);
 for(i=0;i<size;i++)
    scanf("%d",&a[i]);

 big=a[0];
 for(i=1;i<size;i++){
    if(big<a[i])
       big=a[i];
 }
 printf("Maximum element: %d",big);

 small=a[0];
 for(i=1;i<size;i++){
    if(small>a[i])
       small=a[i];
 }
 printf("\n\nMinimum element: %d",small);

 return 0;
}
```

Output

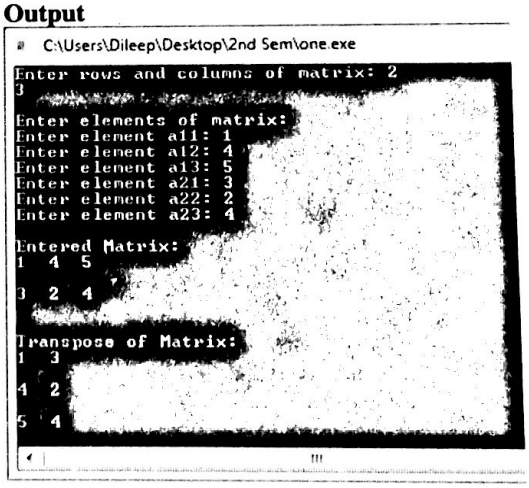


Ques 24) Write a C program to find transpose of a matrix.

Ans: C Program to Find Transpose of a Matrix

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int a[10][10], transpose[10][10], r, c, i, j;
  printf("Enter rows and columns of matrix: ");
  scanf("%d %d", &r, &c);

  // Storing elements of the matrix
  printf("\nEnter elements of matrix:\n");
  for(i=0; i<r; ++i)
    for(j=0; j<c; ++j)
    {
      printf("Enter element a%d%d: ",i+1, j+1);
      scanf("%d", &a[i][j]);
    }

  // Displaying the matrix a[][] */
  printf("\nEntered Matrix: \n");
  for(i=0; i<r; ++i)
    for(j=0; j<c; ++j)
    {
      printf("%d  ", a[i][j]);
      if (j == c-1)
        printf("\n\n");
    }

  // Finding the transpose of matrix a
  for(i=0; i<r; ++i)
    for(j=0; j<c; ++j)
    {
      transpose[j][i] = a[i][j];
    }

  // Displaying the transpose of matrix a
  printf("\nTranspose of Matrix:\n");
  for(i=0; i<c; ++i)
    for(j=0; j<r; ++j)
    {
      printf("%d  ",transpose[i][j]);
      if(j==r-1)
        printf("\n\n");
    }

  return 0;
}
```

Output



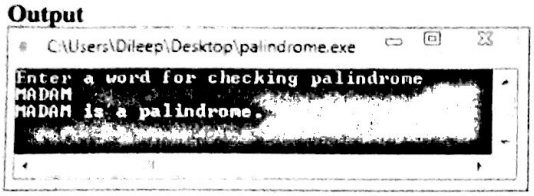
Ques 25) Write a program in C which takes word as input and check it for palindrome.

Ans: C Program to Take a Word as Input and Checking Palindrome

```
#include <stdio.h>
#include <string.h>
int main()
{
     char inputArray[100], reversedArray[100];
     printf("Enter a word for checking palindrome \n");
     scanf("%s", inputArray);
     /* Copy input string and reverse it*/
     strcpy(reversedArray, inputArray);

     /* reverse string */
     strrev(reversedArray);
```

```
    /* Compare reversed string with inpit string */

    if(strcmp(inputArray, reversedArray) == 0 )
        printf("%s is a palindrome.\n", inputArray);
    else
        printf("%s is not a palindrome.\n", inputArray);
    getch();
    return 0; }
```

Output

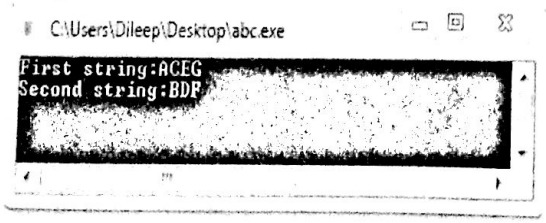


Ques 26) Write a C program which generated two strings from a given string such that all odd positioned characters generate first string and all even positioned characters generate second string. Both generated strings must be stored first and then displayed. For example, "ABCDEFG" should generate two strings "ACEG" and "BDF". Strings should be read and displayed using a single command and no loop should be used for these purposes.

Ans: C Program to Read String and Generates Two Strings using Even and Odd Position Characters

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    char str[] = "ABCDEFG";
    int i=0, j=0, k=0;
    char a[4], b[4];
    while(str[i]!= '\0') {
        if(i%2==0) {
            a[j++] = str[i];
            i++; }
        else {
            b[k++] = str[i]; i++; } }
    printf("First string:");
    j = 0;
    k = 0;
    puts(a);
    printf("Second string:");
    puts(b);
    getch();
    return (0);
}
```

Output

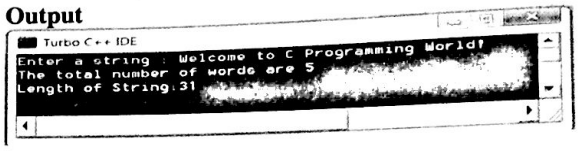


Ques 27) Write a "C" program to count the number of words in a given string and also determine the length of given string.

Ans: C Program to Count the Number of Words and Length of Given String

```
#include<stdio.h>
#include<conio.h>
void main() { char str[50];
    int i,count,countc,len;
    clrscr();
    printf("Enter a string : ");
    gets(str);
    count=0;
    i=0;
    while(str[i]!='\0') {      if(str[i]==' ')
            count++;
            i++; } len=strlen(str);
     printf("The total number of words are %d    ",count+1);
     printf("\nLength of String:%d",len); }
```

Output

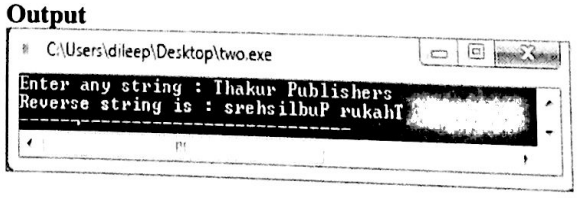


Ques 28) Write a program in C-language to print the reverse of a given string by using standard library function.

Ans: Program to Print the Reverse of a Given String Using Standard Library Function strrev()

```
#include<stdio.h>
#include<string.h>
int main()
{  char str[50];
   printf("Enter any string : ");
   gets(str);
   rev = strrev(str);

   printf("Reverse string is : %s",rev);

   return 0;}
```

Output



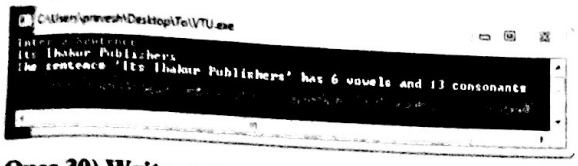
Ques 29) Write a program to read a sentence and count the number of vowels and consonants.

Ans: Program to Read a Sentence and Print the Number of Vowels and Consonants

```
#include <stdio.h>
main()
{    int i, v =0,c=0;
     char str[' '];
     printf("Enter a Sentence\n");
     scanf("%[^\n]s", str);
     for(i=0; str[i]!='\0'; i++) {    switch(str[i]) {      case
     'a':case 'e':case 'i':case 'o':case 'u':case 'A':case 'E':case
     'I':case 'O':case 'U';v++;
```

```
            break;
            default:if(isalpha(str[i]))c++; }}
    printf("The sentence '%s' has %d vowels and %d
    consonants\n",str,v,c);
    getch(); }
```

Output

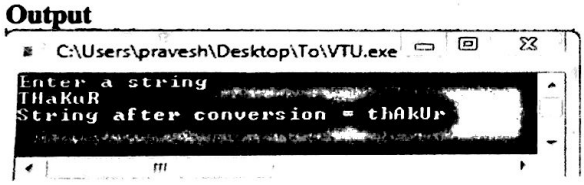


Ques 30) Write a program to convert UPPERCASE alphabets to LOWERCASE alphabets in a given string and vice-versa.

Ans: Program for Converting Uppercase Alphabets to Lowercase Alphabets and Vice-Versa

```
#include <stdio.h>
#include <string.h>
main()
{   int i=1,len;
    char str[30];
    printf("Enter a string\n");
    scanf("%[^\n]s",str);
    len=strlen(str);
    for(i=0; i<len; i++)
    {
        if(str[i]>=97 && str[i]<=122)
        str[i] = str[i] - 32;
        else if(str[i]>=65 && str[i]<=90)str[i] = str[i] + 32;
    }
    printf("String after conversion = %s\n",str);
    getch(); }
```

Output

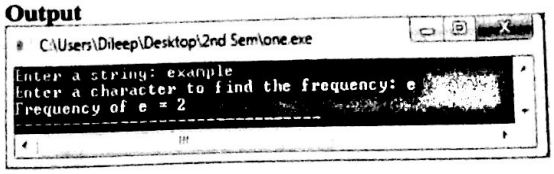


Ques 31) Write a program to find the frequency of characters in a string.

Ans: Program to Find the Frequency of Characters in a String

```
#include<stdio.h>
#include<conio.h>
int main() {
  char str[1000], ch;
  int i, frequency = 0;

  printf("Enter a string: ");
  gets(str);

  printf("Enter a character to find the frequency: ");
  scanf("%c",&ch);

  for(i = 0; str[i] != '\0'; ++i) {
    if(ch == str[i])
      ++frequency;}

  printf("Frequency of %c = %d", ch, frequency);

  return 0; }
```

Output



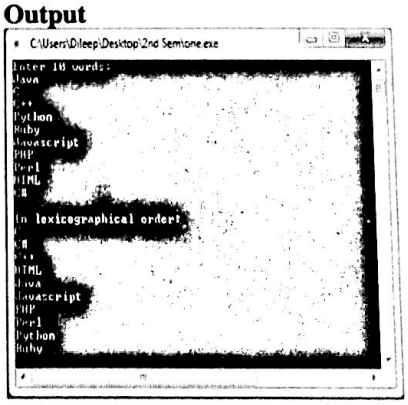
Explanation: In this program, the string entered by the user is stored in variable **str**. Then, the user is asked to enter the character whose frequency is to be found. This is stored in variable **ch**. Now, using the for loop, each character in the string is checked for the entered character. If, the character is found, the **frequency** is increased. If not, the loop continues. Finally, the frequency is printed.

Ques 32) Write a C program to sort elements in lexicographical order (dictionary order).

Ans: Program to Sort Elements in Dictionary Order

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

int main() {
  int i, j;
  char str[10][50], temp[50];

  printf("Enter 10 words:\n");

  for(i=0; i<10; ++i)
    scanf("%s[^\n]",str[i]);

  for(i=0; i<9; ++i)
    for(j=i+1; j<10 ; ++j) {
      if(strcmp(str[i], str[j])>0) {
        strcpy(temp, str[i]);
        strcpy(str[i], str[j]);
        strcpy(str[j], temp);  } }

  printf("\nIn lexicographical order: \n");
  for(i=0; i<10; ++i) {
    puts(str[i]); }

  return 0; }
```

Output



Explanation: Here a two-dimensional string **str** is created. This string can hold maximum of 10 strings and each string can have maximum of 50 characters (including null character). To compare two strings, strcmp() function is used. Also, we used strcpy() function to copy string to a temporary string **temp**.