LAB ASSIGNMENT NO.4

ARRAYS, STRINGS, STRUCTURES AND POINTERS

SHREYAS SAWANT

D2A-55

AIM: Programming using C language

THEORY:

Array: A kind of data structure that can store a fixed-size sequential collection of elements of the same type.
Syntax: type arrayName [arraySize]; (Single Dimension Array)
Declaration: int arr[4];
char arr1[]={'a','b');
int arr[5]={5,4,3,2}
An array element can be accessed by using in general arr_name[i], will return of the ith index of array. There is no out of bounds checking, program compiles but shows unexpected output while running.

String: Strings are actually one-dimensional array of characters terminated by a null character '\0'. Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.
Syntax: char str_name[size]; (an extra terminating character which is the Null character ('\0') used to indicate termination of string which differs strings from normal character arrays.)
Declaration: char str[]={'S','d','\0'}
'&' is not used to take input using scanf.

Structures: Structure is a collection of variables (can be of different types) under a single name.
Syntax: struct [structure tag]
{member definition;
member definition;}obj;
Size of structure is greater than or equal to sum of size of its members. Structure members are accessed through (.) operator. Structure members cannot be initialised within the structure definition. It gives compilation error.

typedef: To give a name to your user defined data types and to give a type a new name. typedef is limited to giving symbolic names to types only whereas #define can be used to define alias for values as well, q., you can define 1 as ONE etc. typedef interpretation is performed by the compiler whereas #definestatements are processed by the pre-processor.

Union: Like Structures, union is a user defined data type. In union, all members share the same memory location.
Syntax: union [union tag]
{member definition;
}obj;
Size of union is equal to size of largest member. Only one member can be accessed and only first member can be initialised. Altering values of any members alters value of other member values.

ARRAY

d) Write a program to display transpose of a matrix and upper and Lower Diagonal

Code:

```
#include <stdio.h>
int n;
void transpose(int [n][n]);
void Utriangle(int [n][n]);
void Ltriangle(int [n][n]);
int main()
{   printf("Enter size of array ");
    scanf("%d",&n);
    int a[n][n];
    printf("Enter elements of array ");
    for(int i=0;i<n;i++)
    {   for(int j=0;j<n;j++)
        {   scanf("%d",&a[i][j]);
        }
    }
    printf("Original matrix\n");
    for(int i=0;i<n;i++)
    {   for(int j=0;j<n;j++)
        {   printf("%d\t",a[i][j]);
        }printf("\n");
    }
    transpose(a);
    Utriangle(a);
    Ltriangle(a);
    return 0;
}
void transpose(int b[n][n])
{ printf("Transpose matrix\n");
  int c[n][n];
    for(int i=0;i<n;i++)
    {   for(int j=0;j<n;j++)
        {   c[i][j]=b[j][i];
        }
    }
    for(int i=0;i<n;i++)
    {   for(int j=0;j<n;j++)
        {   printf("%d\t",c[i][j]);
        }printf("\n");
    }
}
void Utriangle(int b[n][n])
{   printf("\nUpper Triangle\n");
    for(int i=0;i<n;i++)
    {   for(int j=0;j<n;j++)
        { if(i<=j)
            {printf("%d\t",b[i][j]);}
            else
          printf("0\t");
        }printf("\n");
```

```
   }
}
void Ltriangle(int b[n][n])
{    printf("\nLower Triangle\n");
   for(int i=0;i<n;i++)
   {   for(int j=0;j<n;j++)
      { if(i>=j)
         {printf("%d\t",b[i][j]);}
        else
        printf("0\t");
      }printf("\n");
   }
}
```

Output:

```
Enter size of array 4
Enter elements of array 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Original matrix
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      16

Transpose matrix
1       5       9       13
2       6       10      14
3       7       11      15
4       8       12      16

Upper Triangle
1       2       3       4
0       6       7       8
0       0       11      12
0       0       0       16

Lower Triangle
1       0       0       0
5       6       0       0
9       10      11      0
13      14      15      16
```

STRINGS
b) C program to find the number of vowels, consonants, digits and white space in a string.

Code:

```
#include <stdio.h>
#include <string.h>
int vowels(char []);
int numbers(char []);
int main()
{   char v[]={'a','e','i','o','u','A','E','I','O','U','\0'};

    int v_n,num_n,c_n,w_n=0;
    char s[50];
    printf("Enter a string ");
    gets(s);
    v_n=vowels(s);
    num_n=numbers(s);
    for(int i=0;i<strlen(s);i++)
    { if(s[i]==' ')
       {w_n++;    }
    }
    c_n=strlen(s)-(v_n+num_n+w_n);
    printf("Vowels= %d\nConsonants= %d\nNumbers= %d\nWhitespace= %d",v_n,c_n,num_n,w_n); }

int vowels(char b[50])
{ int k=0;
  char v[]={'a','e','i','o','u','A','E','I','O','U','\0'};
    for(int i=0;i<strlen(b);i++)
    {   for(int j=0;j<10;j++)
        { if(b[i]==v[j])
            k++;
        }   }
    return k;}

int numbers(char b[50])
{ int k=0;
 char num[]={'1','2','3','4','5','6','7','8','9','0'};
    for(int i=0;i<strlen(b);i++)
    {   for(int j=0;j<10;j++)
        { if(b[i]==num[j])
            k++;
        }   }
    return k;}
```

Output:

```
Enter a string I am 18 years old.
Vowels= 5
Consonants= 7
Numbers= 2
Whitespace= 4
```

INPUT:

STRUCTURES
b) C Program to Calculate Difference Between Two Time Periods.

Code:

```
#include <stdio.h>
struct
{
   int h;
   int m;
}t1,t2;
int main()
{ printf("Enter hours and minutes for 1st time period ");
   scanf("%d %d",&t1.h,&t1.m);
  printf("Enter hours and minutes for 2nd time period ");
   scanf("%d %d",&t2.h,&t2.m);
  int dh=abs(t1.h-t2.h);
  int dm=abs(t1.m-t2.m);
  printf("Difference in time period is %d hours %d minutes",dh,dm);
}
```

Output:

```
Enter hours and minutes for 1st time period 4 56
Enter hours and minutes for 2nd time period 8 16
Difference in time periods is 4 hours 40 minutes
```

POINTERS
b) WAP to add the elements of the array .

Code:

```
#include <stdio.h>
int main()
{   int n;
    printf("Enter number of elements ");
    scanf("%d",&n);
    int a[n];
    printf("\nEnter elements of array ");
    for(int i=0;i<n;i++)
        scanf("%d",&a[i]);
    int s=0,*p;
    printf("\nSum of elements ");
    for(p=&a[0];p<=&a[n-1];*p++)
        s+=*p;
    printf("%d",s);
    return 0;
}
```

Output:

```
Enter number of elements 7
Enter elements of array 1 2 3 4 5 6 7

Sum of elements 28
```