SYSTEM PROGRAMMING LABORATORY (CSX-326)

LAB PRACTICALS RECORD

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR – 144011, PUNJAB (INDIA)
January - June, 2016

Submitted To:

Mrs. Kamalpreet Kaur
Asst. Professor
Dept. of CSE

Submitted By:

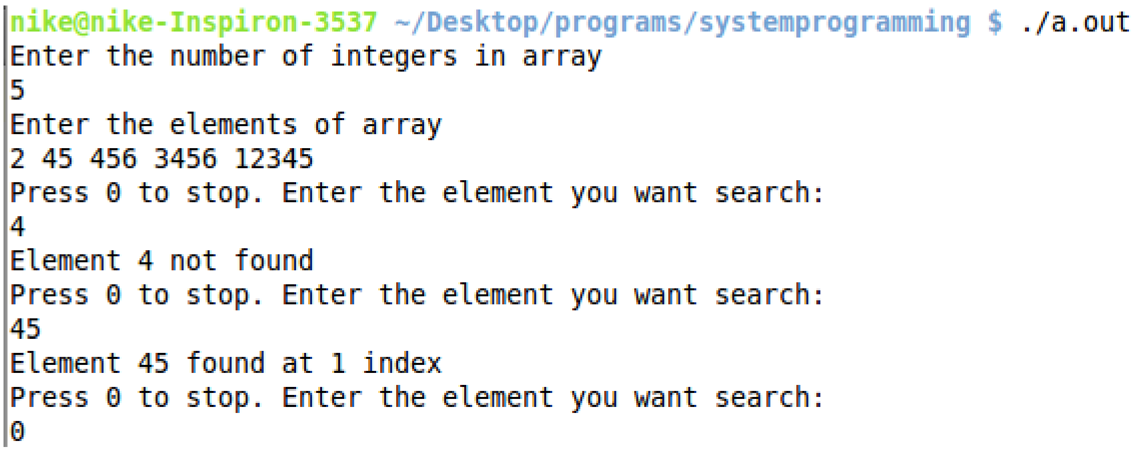
Nikhil Bansal
13103011
6th semester

1.	**Objective:-To implement Linear search.**

Program:

```
#include<bits/stdc++.h>
using namespace std;

int linearsearch(int array[],int start, int end, int search)
{
        int i;
        for(i=start;i<=end;i++){
                if(array[i]==search)
                        break;
        }
        if(i<=end)
                return i;
        else
                return -1;
}

int main()
{
        int array[1000],n,search,i;

        //input of array
        printf("Enter the number of integers in array\n");
        scanf("%d",&n);
        printf("Enter the elements of array\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&array[i]);
        }

        //get query and output the result
        while(1){
                printf("Press 0 to stop. Enter the element you want search:\n");
                scanf("%d",&search);
                if(search==0)
                        break;

                i = linearsearch(array,0,n-1,search);

                if(i==-1)
                        printf("Element %d not found\n",search);
                else
                        printf("Element %d found at %d index\n",search,i);
        }
        return 0;
}
```

Output:



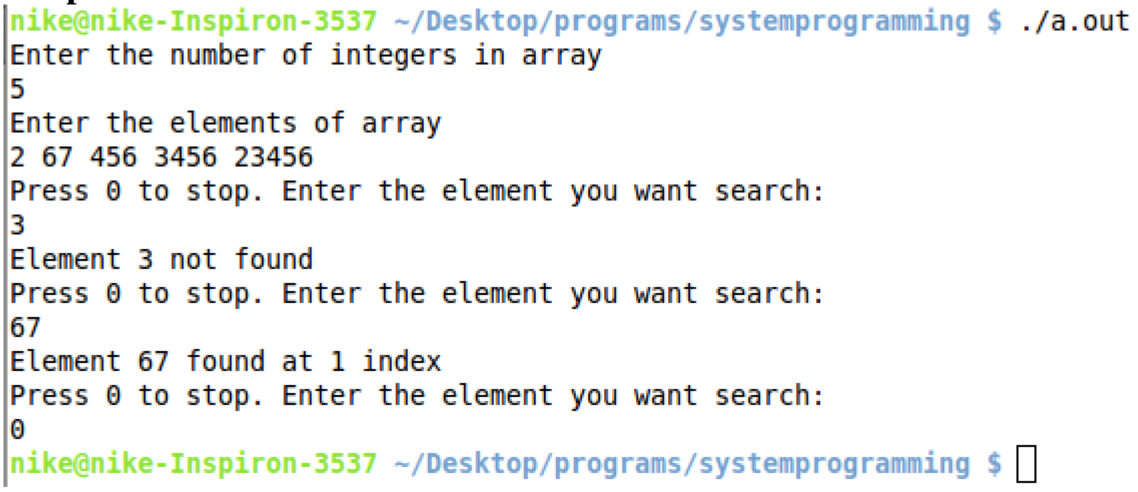
```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the number of integers in array
5
Enter the elements of array
2 45 456 3456 12345
Press 0 to stop. Enter the element you want search:
4
Element 4 not found
Press 0 to stop. Enter the element you want search:
45
Element 45 found at 1 index
Press 0 to stop. Enter the element you want search:
0
```

2.	**Objective:-To implement Binary search.**

Program:

```
#include<bits/stdc++.h>
using namespace std;

int binarysearch(int array[],int start, int end, int search)
{
        if(start>end)
                return -1;

        int mid=(start+end)/2;

        if(array[mid]==search)
                return mid;
        else if(array[mid]>search)
                return binarysearch(array, start, mid-1, search);
        else
                return binarysearch(array, mid+1, end, search);
}

int main()
{
        int array[1000],n,search,i;

        //input of array
        printf("Enter the number of integers in array\n");
        scanf("%d",&n);
        printf("Enter the elements of array\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&array[i]);
        }

        //sort if the array is not sorted
        sort(array, array+n);

        //input query and display output
        while(1){
                printf("Press 0 to stop. Enter the element you want search:\n");
                scanf("%d",&search);
                if(search==0)
                        break;

                i = binarysearch(array,0,n-1,search);

                if(i==-1)
                        printf("Element %d not found\n",search);
                else
                        printf("Element %d found at %d index\n",search,i);
        }
        return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the number of integers in array
5
Enter the elements of array
2 67 456 3456 23456
Press 0 to stop. Enter the element you want search:
3
Element 3 not found
Press 0 to stop. Enter the element you want search:
67
Element 67 found at 1 index
Press 0 to stop. Enter the element you want search:
0
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ 
```

3.	**Objective:-To implement Merge Sort**

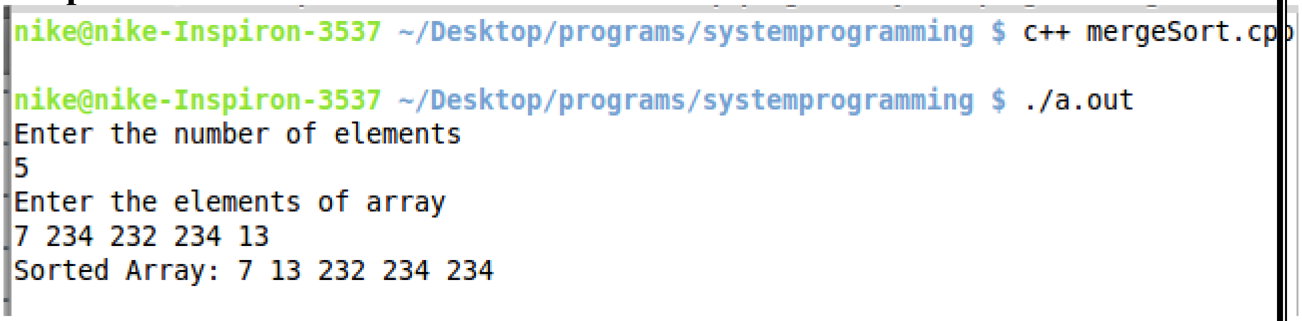
Program:

```
#include<bits/stdc++.h>
using namespace std;

//print function to display array
int print(int array[],int start,int end){
        int i;
        for(i=start;i<=end;i++){
                printf("%d ",array[i]);
        }
        printf("\n");
        return 0;
}

//returns the copy of array elemets
int* copy(int array[],int start,int end){
        int i,*copy_arr=(int *)malloc(sizeof(int)*(end-start+1));
        for(i=start;i<=end;i++){
                copy_arr[i-start]=array[i];
        }
        return copy_arr;
}

//merge the 2 sorted arrays
int merge(int array[],int start,int end,int firsthalf[],int secondhalf[]){

        int i=0,j=0,mid=(start+end)/2,sizefirst=mid-start+1,sizesecond=end-mid;
        int k=0,size=end-start;

        while(k!=size){
                if(i==sizefirst||j==sizesecond)
                        break;
                if(firsthalf[i]>secondhalf[j]){
                        array[start+k]=secondhalf[j];
                        j++;
                }
                else{
                        array[start+k]=firsthalf[i];
                        i++;
                }
                k++;
        }

        while(i<sizefirst){
                array[start+k]=firsthalf[i];
                i++;
                k++;
        }
```

```
	while(j<sizesecond){
		array[start+k]=secondhalf[j];
		j++;
		k++;
	}
	return 0;
}

//main merge sort function
int mergesort(int array[],int start,int end){
	if(start>=end)
		return 0;
	int mid;
	mid=(start+end)/2;

	//sort the 2 sub-arrays
	mergesort(array,start,mid);
	mergesort(array,mid+1,end);

	//create the copy of 2 subarrays
	int *firsthalf=copy(array,start,mid);
	int *secondhalf=copy(array,mid+1,end);

	// merge them in the main array
	merge(array,start,end,firsthalf,secondhalf);

	//free the malloc memory allocated for copies.
	free(firsthalf);
	free(secondhalf);
	return 0;
}

int main()
{
	int n,i,temp,array[1000];

	//input
	printf("Enter the number of elements\n");
	scanf("%d",&n);
	printf("Enter the elements of array\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&array[i]);
	}

	//sort
	mergesort(array,0,n-1);

	//output array
	printf("Sorted Array: ");
	print(array,0,n-1);
```

```
        printf("\n");
        return 0;
}
```

Output:



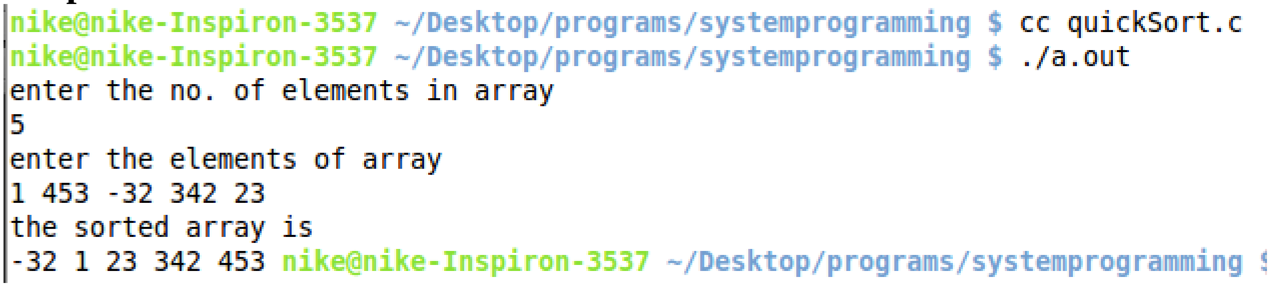
```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ c++ mergeSort.cpp

nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the number of elements
5
Enter the elements of array
7 234 232 234 13
Sorted Array: 7 13 232 234 234
```

4.	**Objective:-To implement QuickSort**

Program:

```
#include<stdio.h>
#include<malloc.h>

//partition function to divide the arrary about a pivot
int partition(int *a,int i,int j){
    int v=j,u=i+1,w;
    while(v>=u){
        for(;u<=j;u++){
            if(a[u]>a[i])
               break;
         }
        for(;v>=i;v--){
            if(a[v]<a[i])
                break;
         }
        if(v>u){
          w=a[u];
          a[u]=a[v];
          a[v]=w;
        }
    }
    if(v==i-1){
        return u-1;
    }
    else{
        return v;
    }
}

//main quick sort function
int quicksort(int *a,int i,int j){
    if(i>=j)
        return 0;
    int u,w;
    u=partition(a,i,j);
    w=a[i];
    a[i]=a[u];
    a[u]=w;
    quicksort(a,i,u-1);
    quicksort(a,u+1,j);
    return 0;
}

int main(){
    int n,*a,i,s;

    //input
    printf("enter the no. of elements in array\n");
    scanf("%d",&n);
    a=(int *)malloc(sizeof(int)*n);

    printf("enter the elements of array\n");
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }

    //sort
    quicksort(a,0,n-1);

    //print the sorted array
    printf("the sorted array is \n");
    for(i=0;i<n;i++){
        printf("%d ",a[i]);
    }

    return 0;
}
```

Output:



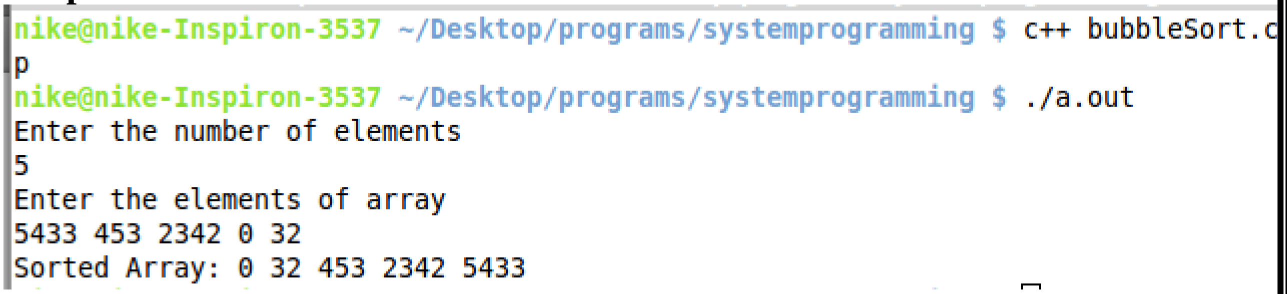
```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ cc quickSort.c
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
enter the no. of elements in array
5
enter the elements of array
1 453 -32 342 23
the sorted array is
-32 1 23 342 453 nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming
```

5.	Objective:-To implement Bubble Sort

Program:

```
#include<bits/stdc++.h>
using namespace std;

int bubblesort(int array[],int start,int end)
{
	int i,j,temp;
	for(i=start;i<=end;i++)
	{
		for(j=start;j<end-i;j++)
		{
			if(array[j]>array[j+1])
			{
				temp=array[j];
				array[j]=array[j+1];
				array[j+1]=temp;
			}
		}
	}
	return 0;
}

int main()
{
	int n,i,temp,array[1000];

	//input
	printf("Enter the number of elements\n");
	scanf("%d",&n);
	printf("Enter the elements of array\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&array[i]);
	}

	//sort
	bubblesort(array,0,n-1);

	//print output array
	printf("Sorted Array: ");
	for(i=0;i<n;i++)
	{
		printf("%d ",array[i]);
	}
	printf("\n");
	return 0;
}
```

Output:



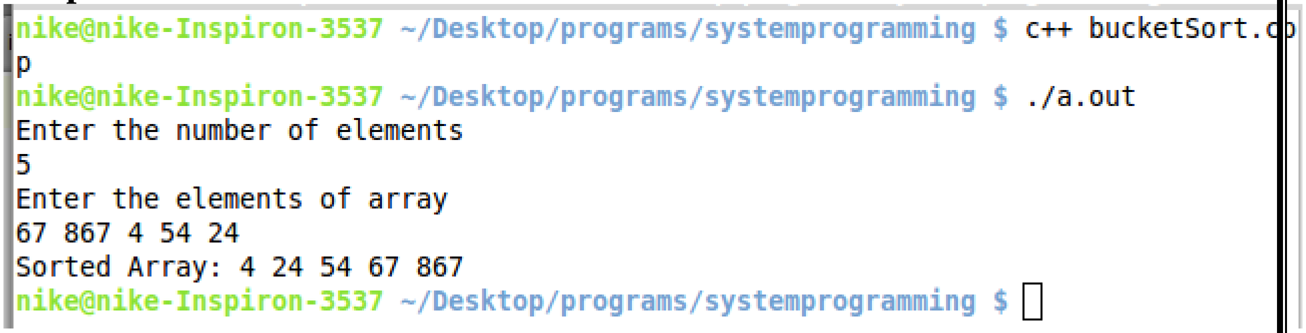
```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ c++ bubbleSort.c
p
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the number of elements
5
Enter the elements of array
5433 453 2342 0 32
Sorted Array: 0 32 453 2342 5433
```

6.	**Objective:-To implement Bucket Sort**

Program:

```
#include<bits/stdc++.h>
using namespace std;
#define NO_OF_BUCKET 100000
#define BUCKET_SIZE 1000

int bucketsort(int array[],int start,int end){
	vector<int> buckets[NO_OF_BUCKET];
	int i,j,k,size;

	//push the elements of array to appropriate bucket
	for(i=start;i<=end;i++){
		buckets[array[i]/BUCKET_SIZE].push_back(array[i]);
	};

	//sort each bucket by inbuilt sort function
	for(i=0;i<NO_OF_BUCKET;i++){
		sort(buckets[i].begin(),buckets[i].end());
	}

	//update the array with sorted array
	k=start;
	for(i=0;i<NO_OF_BUCKET;i++){
		size=buckets[i].size();
		for(j=0;j<size;j++){
			array[k]=buckets[i][j];
			k++;
		}
	}

	return 0;
}

int main()
{
	int n,i,temp,array[1000];
	int arrayint[1000];

	//input
	printf("Enter the number of elements\n");
	scanf("%d",&n);
	printf("Enter the elements of array\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&arrayint[i]);
		array[i]=arrayint[i]*10000;
	}

	//sort
	bucketsort(array,0,n-1);

	//print sorted array
	printf("Sorted Array: ");
	for(i=0;i<n;i++)
	{
		arrayint[i]=1.00*array[i]/10000;
		printf("%d ",arrayint[i]);
	}
	printf("\n");

	return 0;
}
```

Output:



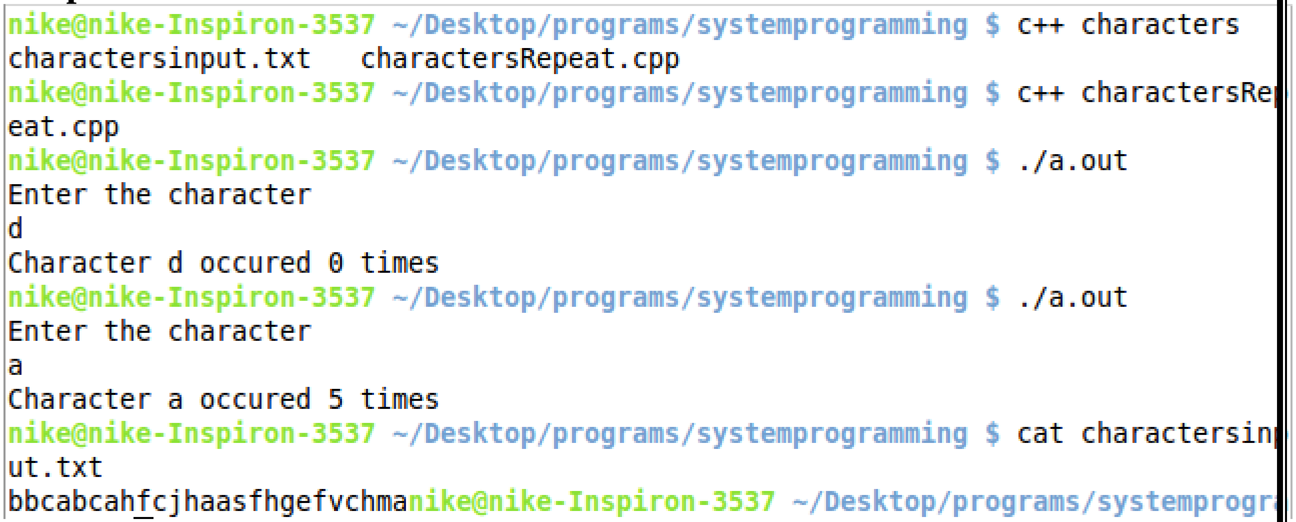
```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ c++ bucketSort.cp
p
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the number of elements
5
Enter the elements of array
67 867 4 54 24
Sorted Array: 4 24 54 67 867
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $
```

7.	**Objective:-To find frequency of a letter in text**

Program:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
	FILE *input = fopen("charactersinput.txt", "r");
	char temp, search;
	int ans=0;
	printf("Enter the character\n");
	scanf("%c",&search);
	while((temp=fgetc(input))!=EOF)
	{
		if(temp==search)
			ans++;
	}
	printf("Character %c occured %d times\n",search,ans);
	return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ c++ characters
charactersinput.txt   charactersRepeat.cpp
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ c++ charactersRe
eat.cpp
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the character
d
Character d occured 0 times
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ ./a.out
Enter the character
a
Character a occured 5 times
nike@nike-Inspiron-3537 ~/Desktop/programs/systemprogramming $ cat charactersin
ut.txt
bbcabcahfcjhaasfhgefvchmanike@nike-Inspiron-3537 ~/Desktop/programs/systemprogr
```

8.	**Objective:-Text Editor**

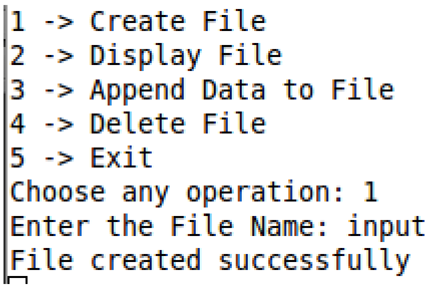
Program:

```
#include<bits/stdc++.h>
using namespace std;

// display the contens of a file
// returns -1 if doesnot exist
int displayFile(char filename[]){
	FILE *myFile=fopen(filename, "r");
	if(myFile==NULL)
		return -1;
	char c;
	while((c=fgetc(myFile))!=EOF){
		printf("%c",c);
	}
	fclose(myFile);
	return 0;
}

// create the file
// returns -1 if creation failed
int createFile(char filename[]){
	FILE *myFile=fopen(filename, "w");
	if(myFile==NULL)
		return -1;
	fclose(myFile);
	return 0;
}

// append the contents in file
int appendDataToFile(char filename[], char data[]){
	FILE *myFile=fopen(filename, "a");
	if(myFile==NULL)
		return -1;
	for(int i=0;data[i]!='\0';i++){
		fputc(data[i], myFile);
	}
	fclose(myFile);
	return 0;
}

// delete the contents in file
int deleteFile(char filename[]){
	int x=remove(filename);
	return x;
}

// get data to append
// stops the input data once :q is received
void inputDataToAppend(char data[]){
	printf("Enter the data.\nPress :q to stop entering data.\n");
	int flag1=0, flag2=0, top=0;
	char buffer;
	// for redundant new line character that gets received.
	scanf("%c",&buffer);
	while(top<1000){
		scanf("%c",&buffer);
		if(buffer==':'){
			flag1=1;
		}
		else if(buffer=='q'){
			flag2=1;
		}
		else{
			flag1=0;
			flag2=0;
		}
		if(flag1==1&&flag2==1){
			data[top-1]='\0';
			break;
		}
		data[top]=buffer;
		top++;
	}
	if(top==1000){
		printf("Buffer Overflow\n");
	}
}
int main(int argc, char const *argv[])
{
	char filename[100],data[1000];
	int operation,err;
	while(1){
		system("sleep 2");
		system("clear");
		printf("1 -> Create File\n2 -> Display File\n3 -> Append Data to File\n4 -> Delete
File\n5 -> Exit\n");
		printf("Choose any operation: ");
		scanf("%d",&operation);
		switch(operation){
			case 1:
				printf("Enter the File Name: ");
				scanf("%s",filename);
				err=createFile(filename);
				if(err==-1)
					printf("File cannot be created\n");
				else
					printf("File created successfully\n");
				break;
			case 2:
				printf("Enter the File Name: ");
```

```
                        scanf("%s",filename);
                        err=displayFile(filename);
                        if(err==-1)
                                printf("File cannot be displayed\n");
                        break;
                case 3:
                        printf("Enter the File Name: ");
                        scanf("%s",filename);
                        inputDataToAppend(data);
                        appendDataToFile(filename, data);
                        printf("\nData appended to File: %s\n",filename);
                        break;
                case 4:
                        printf("Enter the File Name: ");
                        scanf("%s",filename);
                        err=deleteFile(filename);
                        if(err==-1)
                                printf("File cannot be deleted\n");
                        else
                                printf("File deleted successfully\n");
                        break;
                case 5:
                        printf("Exiting.....\n");
                        exit(0);
                        break;
                default:
                        printf("Enter a Valid Option\n");
        }
    }

    return 0;
}
```

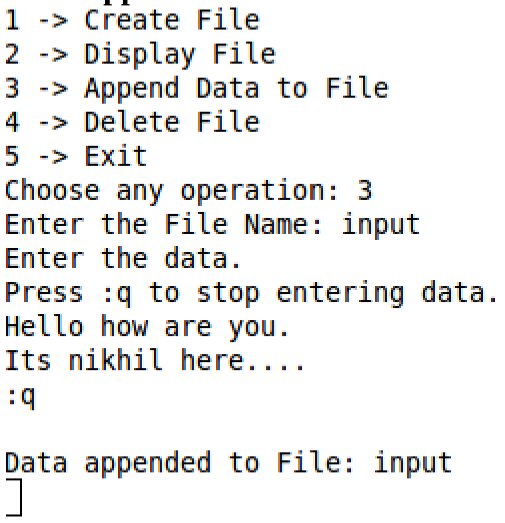
Output:

File Creation:



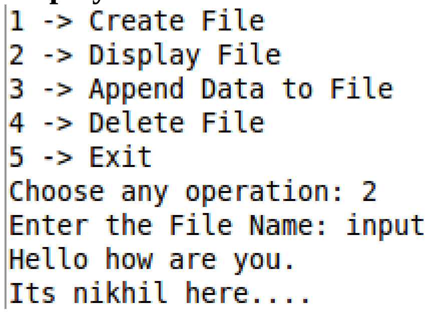
```
1 -> Create File
2 -> Display File
3 -> Append Data to File
4 -> Delete File
5 -> Exit
Choose any operation: 1
Enter the File Name: input
File created successfully
```

Data Append:



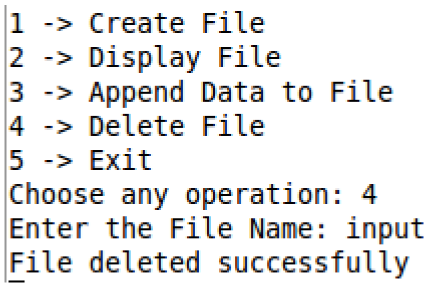
```
1 -> Create File
2 -> Display File
3 -> Append Data to File
4 -> Delete File
5 -> Exit
Choose any operation: 3
Enter the File Name: input
Enter the data.
Press :q to stop entering data.
Hello how are you.
Its nikhil here....
:q

Data appended to File: input
```

Display File:



```
1 -> Create File
2 -> Display File
3 -> Append Data to File
4 -> Delete File
5 -> Exit
Choose any operation: 2
Enter the File Name: input
Hello how are you.
Its nikhil here....
```

Delete File:



```
1 -> Create File
2 -> Display File
3 -> Append Data to File
4 -> Delete File
5 -> Exit
Choose any operation: 4
Enter the File Name: input
File deleted successfully
```

10.	**Objective:- Write a program to make a two Pass-Assembler**

Program:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
        FILE *fp,*f1,*f2;
        char arr[10], temp[10];
        int value[]={1432,1324,3234,1998,2734,1256};
        int count=ORG,arg1,val,i;
        fp=fopen("input.in","r");
        f1=fopen("intermediate.out","w");
        f2=fopen("output.out","w");
        char inst[][10]={"LDA","MOV","ADD","STA","HLT"};
        int code[]={37,40,80,32,76};
        int len[]={3,3,2,3,1};
        int no;
        while(!feof(fp))
        {
                fprintf(f1,"%d ",count);
                fprintf(f2,"%d ",count);
                fscanf(fp,"%s",arr);
                fscanf(fp,"%d",&no);
                fprintf(f1,"%s ",arr);
                fprintf(f1,"%d\n",no);

                for(i=0;i<5;i++)
                        if(!strcmp(arr,inst[i]))
                        {
                                val=i;
                                break;
                        }
                if(i==4)
                {
                        fprintf(f1,"%s",arr);
                        fprintf(f2,"%d",code[i]);
                        exit(0);
                }
                if(i==5)
                {
                        printf("Invalid Instruction\n");
                        printf("Program about to end\n");
                        exit(0);
                }

                fprintf(f2,"%d ",code[i]);
```

```
            fprintf(f2,"%d\n",no);
            count+=len[i];
        }
        fclose(fp);
        fclose(f1);fclose(f2);
        return 0;
}
```

Input:

```
        LDA 3700
        MOV 52
        LDA 3800
        ADD 25
        STA 3016
        HLT
```

Intermediate:

```
        2000 LDA 3700
        2003 MOV 52
        2006 LDA 3800
        2009 ADD 25
        2011 STA 3016
        2014 HLT 3016
        HLT
```

Output:

```
        2000 37 3700
        2003 40 52
        2006 37 3800
        2009 80 25
        2011 32 3016
        2014 76
```

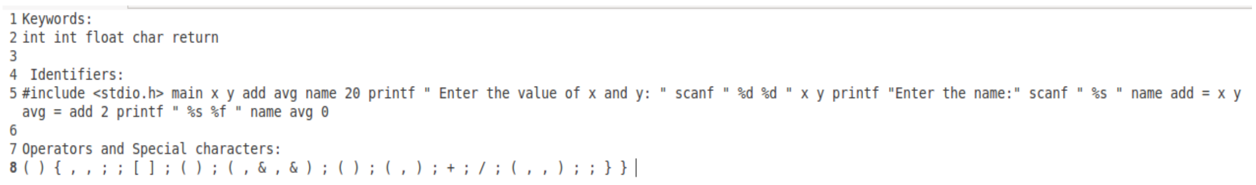
11. Objective:- Write a program for Lexical Analyser

Description:
In computer science, lexical analysis is the process of converting a sequence of characters into a sequence of tokens, i.e. meaningful character strings. A program or function that performs lexical analysis is called a lexical analyzer, lexer, tokenizer, or scanner, though "scanner" is also used for the first stage of a lexer. A lexer is generally combined with a parser, which together analyze the syntax of programming languages, such as in compilers, but also HTML parsers in web browsers, among other examples.
Strictly speaking, a lexer is itself a kind of parser – the syntax of some programming languages is divided into two pieces: the lexical syntax (token structure), which is processed by the lexer; and the phrase syntax, which is processed by the parser. The lexical syntax is usually a regular language, whose alphabet consists of the individual characters of the source code text. The phrase syntax is usually a context-free language, whose alphabet consists of the tokens produced by the lexer. While this is a common separation, alternatively, a lexer can be combined with the parser in scannerless parsing.

Program:
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define ARRAY_SIZE(a) sizeof(a)/sizeof(a[0])
#define ALPHABET_SIZE (26)
#define CHAR_TO_INDEX(c) ((int)c - (int)'a')
typedef struct trie_node trie_node_t;
struct trie_node{
	int value;
	trie_node_t *children[ALPHABET_SIZE];
};
typedef struct trie trie_t;
struct trie{
	trie_node_t *root;
	int count;
};
trie_node_t *getNode(void){
	trie_node_t *pNode = NULL;
	pNode = (trie_node_t *)malloc(sizeof(trie_node_t));
	if( pNode )    {
	int i;
	pNode->value = 0;
	for(i = 0; i < ALPHABET_SIZE; i++){
		pNode->children[i] = NULL;
	}
	}
	return pNode;
}
int initialize(trie_t *pTrie){
   pTrie->root = getNode();
   pTrie->count = 0;
           return 0;
}
int insert(trie_t *pTrie, char key[]){
           int level;
           int length = strlen(key);
           int index;
           trie_node_t *pCrawl;
           pTrie->count++;
           pCrawl = pTrie->root;
           for( level = 0; level < length; level++ ){
           index = CHAR_TO_INDEX(key[level]);
           if( !pCrawl->children[index]){
                     pCrawl->children[index] = getNode();
           }
           pCrawl = pCrawl->children[index];
           }
           pCrawl->value = pTrie->count;
           return 0;
}
int search(trie_t *pTrie, char key[]){
           int level;
           int length = strlen(key);
           int index;
           trie_node_t *pCrawl;
           pCrawl = pTrie->root;
           for( level = 0; level < length; level++ ){
           index = CHAR_TO_INDEX(key[level]);
                     if(!(key[level]<='z'&&key[level]>='a')){
                               return 0;
                     }
           if( !pCrawl->children[index] ){
                     return 0;
           }
       pCrawl = pCrawl->children[index];
           }
           return (0 != pCrawl && pCrawl->value);
}
int lsearch(char o[][9],char c[],int s){
           int i;
           for(i=0;i<s;i++)
                     if(strcmp(o[i],c)==0)
                               break;
           if(i==s)
                     return -1;
           else
                     return i+1;
}
```

```
int main(){
	FILE *f1,*f2;
	f1 = fopen("input.c","r");
	f2 = fopen("output.txt","w");
	char keys[32][9] = {"auto", "break", "case", "char", "const", "continue", "default",
"do","double","else","enum","extern","float","for","goto","if","int",

"long","register","return","short","sizeof","signed","static","struct","switch","typedef","union","unsigned",
"void","volatile","while" };
	trie_t trie;
	char opers[][9] = {"(" , ")" , "{" , "}" , ";" , "," , "[" , "]" , "&" , "+", "/"};
	int size = 11;
	initialize(&trie);
	int i,k=0,o=0,id=0;
	for(i = 0; i < 32; i++){
	insert(&trie, keys[i]);
	}
	char c[9],keyw[100][9],oper[100][5],idnt[100][100];
	while(!feof(f1)){
		fscanf(f1,"%s",c);
		if(search(&trie, c)==1)
			strcpy(keyw[k++],c);
		else if(lsearch(opers,c,size)!=-1)
			strcpy(oper[o++],c);

		else
			strcpy(idnt[id++],c);



	}
	fprintf(f2,"Keywords:\n");
	for(i=0;i<k;i++){
		fprintf(f2,"%s ",keyw[i]);
	}
	fprintf(f2,"\n \n Identifiers:\n");
	for(i=0;i<id;i++){
		fprintf(f2,"%s ",idnt[i]);
	}
	fprintf(f2,"\n \nOperators and Special characters:\n");
	for(i=0;i<o;i++){
		fprintf(f2,"%s ",oper[i]);
	}
	fclose(f1);
	fclose(f2);
	return 0;
}
```

Output:



```
Keywords:
int int float char return

 Identifiers:
#include <stdio.h> main x y add avg name 20 printf " Enter the value of x and y: " scanf " %d %d " x y printf "Enter the name:" scanf " %s " name add = x y
avg = add 2 printf " %s %f " name avg 0

Operators and Special characters:
( ) { , , ; ; [ ] ; ( ) ; ( , & , & ) ; ( ) ; ( , ) ; + ; / ; ( , , ) ; ; } }
```