**PRACTICAL – 1**

**Write a program to implement Linear Search using Array**.

#include <stdio.h>

int main() {

int s,num,i,flag=0;

printf("Enter number of elements in array");

scanf("%d",&s);

int arr[s];

printf("Enter the elements of Array");

for(i=0;i<s;i++){

scanf("%d",&arr[i]);

}

printf("Enter the element to be searched");

scanf("%d",&num);

for(i=0;i<s;i++){

if(arr[i]==num){

flag=1;

break;

}

}

if(flag==1){

printf("Element found in the array at postition %d",i+1);

}

else{

printf("Element is not present in the array \n");

}

return 0;

}

**OUTPUT :**

**Case 1:** Element is not found.

**Case 2:** Element is found.

**PRACTICAL – 2**

**Write a program to implement Binary Search using Array**.

#include <stdio.h>

int main()

{

int i,j,low,high,middle,n,search,arr[200];

printf("Enter no. of elements");

scanf("%d",&n);

printf("Enter elements of array in sorted form :");

for(i=0;i<n;i++){

scanf("%d",&arr[i]);

}

printf("Enter value to be searched :");

scanf("%d",&search);

low=0;

high=n-1;

middle=(low+high)/2;

while(low<=high){

if(search==arr[middle]){

printf("%d is found at location %d ",search,middle+1);

break;

}

else if(search>arr[middle]){

low=middle+1;

}

else{

high = middle-1;

middle=(low+high)/2;

}

}

if(low>high){

printf("%d is not present in the array ",search);

}

return 0;

}

**PRACTICAL – 3**

**Write a program to implement Bubble Sort using Array**.

#include<stdio.h>

#include<conio.h>

void bubblesort();

int a[100],n,i,j,temp;

void main()

{

printf("Input the number of elements in the array :\n");

scanf("%d",&n);

printf("Input %d elements in the array \n",n);

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

bubblesort();

getch();

}

void bubblesort()

{

for(i=1;i<n;i++)

{

for(j=0;j<n-i;j++)

{

if(a[j]>a[j+1])

{

temp=a[j];

a[j] = a[j+1];

a[j+1]=temp;

}

}

}

printf("\n Array after sorting is : \n\n");

for(i=0;i<n;i++)

{

printf("%d\n",a[i]);

}

}

**PRACTICAL – 4**

**Write a program to implement Insertion Sort using Array**.

#include<stdio.h>

#include<conio.h>

void isort();

int a[100],n,i,j,temp;

void main()

{

clrscr();

printf("Input number of elements of array :\n");

scanf("%d",&n);

printf("Input %d elements in the array : \n\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

isort();

getch();

}

void isort()

{

for(i=1;i<n;i++)

{

temp=a[i];

j=i-1;

while(j>=0&&a[j]>temp)

{

a[j+1]=a[j];

j--;

}

a[j+1]=temp;

}

printf("Your array after sorting :\n\n");

for(i=0;i<n;i++)

{

printf("%d\n",a[i]);

}

}

**PRACTICAL – 5**

**Write a program for implementation of stack functions :push,pop,display**.

#include <stdio.h>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

top=-1;

printf("\n Enter the size of STACK[MAX=100] :");

scanf("%d",&n);

printf("Stack operation using array");

printf("\n\t 1.push \n 2.pop \n 3.display \n4.exit");

do{

printf("\n Enter the choice");

scanf("%d",&choice);

switch(choice){

case 1:{

push();

break;

}

case 2:{

pop();

break;

}

case 3:{

display();

break;

}

case 4:{

printf("\n\t Exit points");

break;

}

default:

{

printf("Enter a valid choice");

}

}

}

while(choice!=4);

return 0;

}

void push()

{

if(top>=n-1)

{

printf("Stack is overflow");

}

else

{

printf("Enter a value to be pushed");

scanf("%d",&x);

top++;

stack[top]=x;

}

}

void pop()

{

if(top<=-1){

printf("Stack is under flow");

}

else

{

printf("The poped element is %d ",stack[top]);

top--;

}

}

void display()

{

if(top>=0)

{

printf("The element in STACK\n");

for(i=top;i>=0;i--)

{

printf("\n%d",stack[i]);

}

}

else

{

printf("The stack is Empty");

}

}

**PRACTICAL – 6(a)**

**Write a program to find factorial of a number using recursion.**

#include <stdio.h>

int factorial(int n)

{

if(n>=1){

return n\*factorial(n-1);

}

else

{

return 1;

}

}

int main(){

int n;

printf("Enter the number to find facorial");

scanf("%d",&n);

printf("The factorial of %d is %d ",n,factorial(n));

return 0;

}

**PRACTICAL – 6(b)**

**Write a program to make Fibonacci series of n terms using recursion.**

#include <stdio.h>

void fib(int n,int a,int b)

{

if(n-2!=0)

{

int c;

c=a+b;

a=b;

b=c;

n--;

printf("%d \t",c);

fib(n,a,b);

}

}

int main()

{

int a=0,b=1,n;

printf("Enter the number of terms required in the series : \n");

scanf("%d",&n);

printf("0 1");

fib(n,a,b);

return 0;

}

**PRACTICAL – 6(c)**

**Write a program for Tower of Hanoi using recursion.**

#include <stdio.h>

void TOH(int n,char A,char B,char C)

{

if(n==1)

{

printf("Move Disk 1 from peg %c to peg %c \n",A,B);

}

else

{

TOH(n-1,A,C,B);

printf("Move Disk %d from peg %c to peg %c \n",n,A,B);

TOH(n-1,C,B,A);

}

}

int main()

{

int n;

printf("Enter no. of Disk");

scanf("%d",&n);

printf("\n");

TOH(n,'A','B','C');

return 0;

}

**PRACTICAL – 7**

**Write a program to implement Queue using Array.**

#include <stdio.h>

#include<stdlib.h>

#define MAX 100

int queue[MAX],choice;

void enqueue();

void dequeue();

void display();

int front=-1;

int rear =-1;

int main()

{

while(1){

printf("\n\t 1.Enqueue \n\t 2.Dequeue \n\t 3.Display \n\t 4.Quit");

printf("\n\t Enter Your Choice :\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

default :

printf("Wrong Choice");

}

}

return 0;

}

void enqueue()

{

int item;

if(rear==MAX-1)

{

printf("Queue Overflow\n");

}

else

{

if(front==-1)

{

front=0;

}

printf("Enter the value to be added :\n");

scanf("%d",&item);

rear=rear+1;

queue[rear]=item;

}

}

void display()

{

int i;

if(front==-1)

{

printf("Queue is empty");

}

else

{

printf("Queue is :\n");

for(i=front;i<=rear;i++)

{

printf("%d",queue[i]);

printf("\n");

}

}

}

void dequeue()

{

if(front==-1 || front>rear)

{

printf("Queue Underflow.\n");

return;

}

else

{

printf("Element deleted from queue is : %d\n",queue[front]);

front=front+1;

}

}

**PRACTICAL – 8**

**Write a program to implement Circular Queue using Array.**

#include <stdio.h>

void insertq(int [],int);

void deleteq(int []);

void display(int []);

int front=-1;

int rear=-1;

int choice,n,size;

int main()

{

printf("\nEnter the size of circular queue:");

scanf("%d",&size);

int queue[size];

printf("\n CIRCULAR QUEUE OPERATION USING ARRAY");

printf("\n\t------------------------------------");

printf("\n\t1.Insertion\n\t2.Deletion\n\t3.Display\n\t4.EXIT");

do

{

printf("\n");

printf("Enter the choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

printf("\n Enter number:");

scanf("%d",&n);

insertq(queue,n);

break;

}

case 2:

{

deleteq(queue);

break;

}

case 3:

{

display(queue);

break;

}

case 4:

{

printf("Exit Point");

break;

}

default:

printf("Please Enter a valid choice(1/2/3/4)");

}

}

while(choice!=4);

return 0;

}

void insertq(int queue[],int item)

{

if((front==0 && rear==size-1)||(front==rear+1))

{

printf("Queue is full");

return;

}

else if(rear==-1)

{

rear++;

front++;

}

else if(rear==size-1 && front>0)

{

rear=0;

}

else

{

rear++;

}

queue[rear]=item;

}

void display(int queue[])

{

int i;

printf("\n");

if(front>rear)

{

for(i=front;i<size;i++)

{

printf("%d",queue[i]);

}

for(i=0;i<=rear;i++)

{

printf("%d",queue[i]);

}

}

else

{

for(i=front;i<=rear;i++)

{

printf("%d",queue[i]);

}

}

}

void deleteq(int queue[])

{

if(front==-1)

{

printf("Queue is empty");

}

else if(front==rear)

{

printf("\n%d deleted",queue[front]);

front=-1;

rear=-1;

}

else

{

printf("\n %d deleted",queue[front]);

front++;

}

}

**PRACTICAL – 9**

**Write a program to implement Quick Sort using Array.**

#include <stdio.h>

int partition(int a[],int start,int end)

{

int pivot=a[end];

int i=(start-1);

for(int j=start;j<=end-1;j++)

{

if(a[j]<pivot)

{

i++;

int t=a[i];

a[i]=a[j];

a[j]=t;

}

}

int t=a[i+1];

a[i+1]=a[end];

a[end]=t;

return(i+1);

}

void quick(int a[],int start,int end)

{

if(start<end)

{

int p=partition(a,start,end);

quick(a,start,p-1);

quick(a,p+1,end);

}

}

int main()

{

int size;

printf("Enter the number of elements");

scanf("%d",&size);

int a[size];

printf("Enter elements:");

for(int i=0;i<size;i++)

{

scanf("%d",&a[i]);

}

int n=sizeof(a)/sizeof(a[0]);

quick(a,0,n-1);

printf("After Sorting:");

for(int i=0;i<size;i++)

{

printf("%d",a[i]);

}

return 0;

}

-------------------------------------------------------Practical ---------------------------------------------------------

#include <stdio.h>

void merge(int a[],int beg,int mid,int end)

{

int i,j,k;

int n1=mid-beg+1;

int n2=end-mid;

int LeftArray[n1],RightArray[n2];

for(i=0;i<n1;i++)

{

LeftArray[i]=a[beg+i];

}

for(int j=0;j<n2;j++)

{

RightArray[i]=a[mid+1+j];

}

i=0;

j=0;

k=beg;

while(i<n1 && j<n2)

{

if(LeftArray[i]<=RightArray[j])

{

a[k]=LeftArray[i];

i++;

}

else

{

a[k]=RightArray[j];

j++;

}

k++;

}

while(i<n1)

{

a[k]=LeftArray[i];

i++;

k++;

}

while(j<n2)

{

a[k]=RightArray[j];

j++;

k++;

}

}

void mergeSort(int a[],int beg,int end)

{

if(beg<end)

{

int mid=(beg+end)/2;

mergeSort(a,beg,mid);

mergeSort(a,mid+1,end);

merge(a,beg,mid,end);

}

}

int main()

{

int size;

printf("Enter the no. of elements :");

scanf("%d",&size);

int a[size];

printf("Enter Elements");

for(int i=0;i<size;i++)

{

scanf("%d",&a[i]);

}

int n=sizeof(a)/sizeof(a[0]);

mergeSort(a,0,n-1);

printf("After Sorting");

for(int i=0;i<size;i++)

{

printf("%d\t",a[i]);

}

return 0;

}