1. Write a c++ program to implement the following without using inbuilt functions,

i)Concatenate two strings.

ii)Reverse a given string.

iii)Compare two strings.

iv)Chec whether the given string is present in the main string and return the position.

i)Concatenate two strings.

#include<iostream>

using namespace std;

void strconcat(char str1[10],char str2[10]);

int main()

{

char str1[10],str2[10];

cout<<"Enter the first string:";

cin>>str1;

cout<<"Enter the second string:";

cin>>str2;

strconcat(str1,str2);

return 0;

}

void strconcat(char str1[10],char str2[10])

{

int i=0,j=0;

while(str1[i]!='\0')

{

i++;

}

while(str2[j]!='\0')

{

str1[i]=str2[j];

i++;

j++;

}

str1[i]='\0';

cout<<"The concatinated string is:"<<str1;

}

OUTPUT:

Enter the first string:beaut

Enter the second string:iful

The concatinated string is:beautiful

ii)Reverse a given string.

#include <iostream>

using namespace std;

void strrev(char str1[]);

void strrev(char str1[]) {

int i = 0, j = 0;

char ch;

// Find the length of the string

while (str1[i] != '\0') {

i++;

}

i--; // Adjust to point to the last character of the string

// Reverse the string

while (j < i) {

ch = str1[j];

str1[j] = str1[i];

str1[i] = ch;

j++;

i--;

}

cout << "The reversed string is: " << str1 << endl;

}

int main() {

char str[100]; // Increased size for flexibility

cout << "Enter the string: ";

cin >> str; // Note: This stops at the first whitespace

strrev(str);

return 0;

}

OUTPUT:

Enter the string: ferris

The reversed string is: sirref

iii)Compare two strings.

#include<iostream>

using namespace std;

void strcompare(char str1[10], char str2[10]) {

int i, j, flag = 0, len1 = 0, len2 = 0;

// Calculate lengths of both strings

for (len1 = 0; str1[len1] != '\0'; len1++);

for (len2 = 0; str2[len2] != '\0'; len2++);

if (len1 == len2) {

i = 0;

while (str1[i] != '\0') {

if (str1[i] != str2[i]) {

flag = 1; // Set flag if mismatch is found

break;

}

i++;

}

} else {

flag = 1; // Strings are not equal if lengths differ

}

if (flag == 0) {

cout << "The strings are equal.";

} else {

cout << "The strings are not equal.";

}

}

int main() {

char str1[10], str2[10];

cout << "Enter the first string: ";

cin >> str1;

cout << "Enter the second string: ";

cin >> str2;

strcompare(str1, str2);

return 0;

}

OUTPUT:

Enter the first string: demo

Enter the second string: demo

The strings are equal.

iv)Check whether the given string is present in the main string and return the position.

#include <iostream>

using namespace std;

int strpos(char str1[20], char str2[20]) {

int pos = -1, i = 0, j = 0;

while (str1[j] != '\0') {

// Match characters of str2 with str1

if (str2[i] == str1[j]) {

if (pos == -1) {

pos = j; // Mark the starting position

}

i++;

if (str2[i] == '\0') {

return pos; // Match found

}

} else {

if (pos != -1) {

j = pos; // Reset j to the position after the last match attempt

}

i = 0;

pos = -1;

}

j++;

}

return -1; // No match found

}

int main() {

char str1[20], str2[20];

int pos;

// Input strings

cout << "Enter a string (max 19 characters): ";

cin >> str1;

cout << "Enter the string to search (max 19 characters): ";

cin >> str2;

// Search for substring

pos = strpos(str1, str2);

// Output result

if (pos == -1) {

cout << "String not found!";

} else {

cout << "String found at position: " << pos + 1; // Convert to one-based position

}

return 0;

}

OUTPUT:

Enter a string (max 19 characters): honeybee

Enter the string to search (max 19 characters): ney

String found at position: 3

--------------------------------------------------------------------------------------------------

2.Write a c++ program to implement Stack data structure using arrays and class.

#include<iostream>

#define MAX 5

using namespace std;

class Stack

{

int stk[MAX],top;

public:

Stack()

{

top=-1;

}

void push();

void pop();

void sdisplay();

};

int main()

{

Stack s;

int ch;

cout<<"Menu\n1.Push\n2.Pop\n3.Display\n4.Exit\n";

while(1)

{

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:s.push();

break;

case 2:s.pop();

break;

case 3:s.sdisplay();

break;

case 4:exit(0);

}

}

return 0;

}

void Stack::push()

{

int num;

if(top==MAX-1)

cout<<"Stack is full!\n";

else

{

cout<<"Enter the element:";

cin>>num;

stk[++top]=num;

}

}

void Stack::pop()

{

int num;

if(top==-1)

cout<<"Stack is empty!\n";

else

{

num=stk[top];

top-=1;

cout<<"Popped element is:"<<num<<"";

}

}

void Stack:: sdisplay()

{

int i;

if(top==-1)

cout<<"Stack is empty!\n";

else

{

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

{

cout<<stk[i]<<"\n";

}

}

}

OUTPUT:

Menu

1.Push

2.Pop

3.Display

4.Exit

Enter your choice:1

Enter the element:1

Enter your choice:1

Enter the element:2

Enter your choice:1

Enter the element:3

Enter your choice:1

Enter the element:4

Enter your choice:1

Enter the element:5

Enter your choice:1

Stack is full!

Enter your choice:3

5

4

3

2

1

Enter your choice:2

Popped element is:5

Enter your choice:2

Popped element is:4

Enter your choice:2

Popped element is:3

Enter your choice:2

Popped element is:2

Enter your choice:2

Popped element is:1

Enter your choice:2

Stack is empty!

Enter your choice:3

Stack is empty!

Enter your choice:4

------------------------------------------------------------------------------------------------------------

3.Write a c++ program to convert decimal number to binary number using Stack data structure.

#include<iostream>

#define MAX 20

using namespace std;

class DecToBinary

{

public:

int top;

int stk[MAX];

DecToBinary()

{

top=-1;

}

void push(int obj)

{

stk[++top]=obj;

}

int pop()

{

int obj;

return obj=stk[top--];

}

};

int main()

{

DecToBinary db;

int num,n,rem,binary=0;

cout<<"Enter the number to be converted:";

cin>>num;

n=num;

while(n!=0)

{

rem=n%2;

db.push(rem);

n=n/2;

}

while(db.top!=-1)

{

binary=binary\*10+db.pop();

}

cout<<"Binary value of "<<num<<" is:"<<binary;

return 0;

}

OUTPUT:

Enter the number to be converted:10

Binary value of 10 is:1010

---------------------------------------------------------------------------------------------------

4.Write a c++ program to convert given paranthesized infix expression to postfix expression.

#include<iostream>

using namespace std;

class Convert

{

char infix[30]; //Array to hold the infix expression

char postfix[30]; //Array to hold the postfix expression

char stk[30]; //Stack for operators

int top; //Stack pointer

public:

Convert()

{

top=-1; //Initialize stack pointer

}

void read()

{

cout<<"Enter the infix expression:";

cin>>infix;

}

void display()

{

cout<<"Infix Expression:"<<infix<<"\n";

cout<<"Postfix Expression:"<<postfix;

}

void infpost()

{

int i=0,j=0;

char symb;

while(infix[i]!='\0')

{

symb=infix[i];

//If the symbol is an operand, add it to the postfix expresssion

if(symb>='a'&&symb<='z')

{

postfix[j++]=symb;

}

else if(symb=='(')

{

push(symb); //Push '(' onto the stack

}

else if(symb==')')

{

while(top!=-1&&stk[top]!='(') //Pop from the stack until '(' is found

{

postfix[j++]=pop();

}

pop(); //Remove '(' from the stack

}

else

{

while(top!=-1&&prio(stk[top])>=prio(symb)) //Handling operators

{

postfix[j++]=pop(); //Pop operators of higher or equal precedence

}

push(symb); //Push the current operator onto the stack

}

i++; //Move to the next symbol;

}

while(top!=-1) //Pop all the operators from the stack

{

postfix[j++]=pop();

}

postfix[j]='\0' ; //Null terminate the postfix string

}

int prio(char symb)

{

switch(symb) //Return the precedence of operators

{

case'$':return 3;

case'\*':return 2;

case'/':return 2;

case'+':return 1;

case'-':return 1;

case'(':return 0; //'('has the lowest precedence

case')':return -1; //')' should not appear in the stack

default:return -1; //To handle unexpected characters

}

}

void push(char obj)

{

top+=1;

stk[top]=obj; //Push the object onto the stack

}

char pop()

{

char obj=stk[top];

top--; //Decrease the stack pointer

return obj; //Return the popped object

}

};

int main()

{

Convert c;

c.read(); //Read the infix Expression

c.infpost(); //Convert infix to postfix

c.display(); //isplay the results

return 0;

}

OUTPUT:

Enter the infix expression:(a+b)\*(c+d\*e$(f+g))

Infix Expression:(a+b)\*(c+d\*e$(f+g))

Postfix Expression:ab+cdefg+$\*+\*

----------------------------------------------------------------------------------------------------

5.Write a c++ program to evaluate the given postfix expression using Stack data structure.

#include<iostream>

#include<math.h>

#define MAX 20

using namespace std;

class PostEva

{

int top;

int stk[MAX];

public:

PostEva()

{

top=-1;

}

void push(int obj)

{

stk[++top]=obj;

}

int pop()

{

int obj;

obj=stk[top];

top=top-1;

return obj;

}

int operation(int op1,int op2,char symb)

{

double res=0;

switch(symb)

{

case'+':res=op1+op2;

break;

case'-':res=op1-op2;

break;

case'\*':res=op1\*op2;

break;

case'/':res=op1/op2;

break;

case'^':res=pow(op1,op2);

break;

}

return res;

}

};

int main()

{

PostEva pe;

char postfix[20],symb;

int op,op1,op2,i=0,value=0;

cout<<"Enter the postfix expression(Digits and operators):";

cin>>postfix;

while(postfix[i]!='\0')

{

symb=postfix[i];

if(symb>=48&&symb<=57)

{

op=symb-48;

pe.push(op);

}

else

{

op2=pe.pop();

op1=pe.pop();

value=pe.operation(op1,op2,symb);

pe.push(value);

}

i++;

}

value=pe.pop();

cout<<"The value after Postfix Evaluation:"<<value;

}

OUTPUT:

Enter the postfix expression(Digits and operators):23\*42/+

The value after Postfix Evaluation:8

--------------------------------------------------------------------------------------------------------------

6.Write a c++ program to implement Queue data structure using arrays and class.

#include<iostream>

#define MAX 5

using namespace std;

class Queue

{

int Q[MAX];

int front,rear;

public:

Queue()

{

front=rear=-1;

}

void fninsert()

{

int obj;

if(rear==MAX-1)

cout<<"Queue is full!\n";

else

{

cout<<"Enter the element:";

cin>>obj;

Q[++rear]=obj;

if(front==-1)

front=0;

}

}

void fndelete()

{

int obj;

if(rear==-1)

cout<<"Queue is empty!\n";

else

{

obj=Q[front];

if(front==rear)

front=rear=-1;

else

front=front+1;

cout<<"Deleted element:"<<obj<<"\n";

}

}

void fndisplay()

{

int i;

if(front==-1)

cout<<"Queue is empty!\n";

else

{

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

{

cout<<Q[i]<<"\t";

}

}

}

};

int main()

{

int ch;

Queue q;

cout<<"MENU\n1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n";

while(1)

{

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:q.fninsert();

break;

case 2:q.fndelete();

break;

case 3:q.fndisplay();

break;

case 4:exit(0);

}

}

}

OUTPUT:

MENU

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice:1

Enter the element:1

Enter your choice:1

Enter the element:2

Enter your choice:1

Enter the element:3

Enter your choice:1

Enter the element:4

Enter your choice:1

Enter the element:5

Enter your choice:1

Queue is full!

Enter your choice:3

1 2 3 4 5

Enter your choice:2

Deleted element:1

Enter your choice:2

Deleted element:2

Enter your choice:2

Deleted element:3

Enter your choice:2

Deleted element:4

Enter your choice:2

Deleted element:5

Enter your choice:2

Queue is empty!

Enter your choice:3

Queue is empty!

Enter your choice:4

------------------------------------------------------------------------------------------------------------

7.Write a c++ program to implement Circular Queue data structure using arrays and class.

#include<iostream>

#define MAX 5

using namespace std;

class CQueue

{

int CQ[MAX];

int front,rear;

public:

CQueue()

{

front=rear=-1;

}

void fninsert()

{

int obj;

if((rear+1)%MAX==front)

cout<<"Circular Queue is full!\n";

else

{

cout<<"Enter the element:";

cin>>obj;

rear=(rear+1)%MAX;

CQ[rear]=obj;

if(front==-1)

front=0;

}

}

void fndelete()

{

int obj;

if(front==-1)

cout<<"Circular Queue is empty!\n";

else

{

obj=CQ[front];

if(front==rear)

front=rear=-1;

else

front=(front+1)%MAX;

cout<<"Deleted element:"<<obj<<"\n";

}

}

void fndisplay()

{

int i;

if(front==-1)

cout<<"Circular Queue is empty!\n";

else

{

if(front<=rear)

{

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

cout<<CQ[i]<<"\t";

}

else

{

for(i=front;i<=MAX-1;i++)

cout<<CQ[i]<<"\t";

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

cout<<CQ[i]<<"\t";

}

}

}

};

int main()

{

int ch;

CQueue cq;

cout<<"MENU\n1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n";

while(1)

{

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:cq.fninsert();

break;

case 2:cq.fndelete();

break;

case 3:cq.fndisplay();

break;

case 4:exit(0);

}

}

}

OUTPUT:

MENU

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice:1

Enter the element:1

Enter your choice:1

Enter the element:2

Enter your choice:1

Enter the element:3

Enter your choice:1

Enter the element:4

Enter your choice:1

Enter the element:5

Enter your choice:1

Circular Queue is full!

Enter your choice:

3

1 2 3 4 5

Enter your choice:2

Deleted element:1

Enter your choice:2

Deleted element:2

Enter your choice:2

Deleted element:3

Enter your choice:3

4 5

Enter your choice:1

Enter the element:8

Enter your choice:1

Enter the element:9

Enter your choice:3

4 5 8 9

Enter your choice:2

Deleted element:4

Enter your choice:2

Deleted element:5

Enter your choice:2

Deleted element:8

Enter your choice:2

Deleted element:9

Enter your choice:2

Circular Queue is empty!

Enter your choice:3

Circular Queue is empty!

Enter your choice:4

----------------------------------------------------------------------------------------------------------------------

8.Write a c++ program to implement Input Restricted Double Ended Queue data structure using arrays and class.

#include<iostream>

#define MAX 5

using namespace std;

class DQueue

{

int DQ[MAX];

int front,rear;

public:

DQueue()

{

front=rear=-1;

}

void insertRear()

{

int obj;

if(rear==MAX-1)

cout<<"DeQueue is full!\n";

else

{

cout<<"Enter the element:";

cin>>obj;

DQ[++rear]=obj;

if(front==-1)

front=0;

}

}

void deleteRear()

{

if(front==-1)

cout<<"DeQueue is empty!\n";

else

{

int obj=DQ[rear];

if(front==rear)

front=rear=-1;

else

rear=rear-1;

cout<<"Deleted element:"<<obj<<"\n";

}

}

void deleteFront()

{

if(front==-1)

cout<<"DeQueue is empty!\n";

else

{

int obj=DQ[front];

if(front==rear)

front=rear=-1;

else

front=front+1;

cout<<"Deleted element:"<<obj<<"\n";

}

}

void fndisplay()

{

int i;

if(front==-1)

cout<<"Queue is empty!\n";

else

{

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

{

cout<<DQ[i]<<"\t";

}

}

}

};

int main()

{

int ch;

DQueue dq;

cout<<"MENU\n1.Insert at the Rear\n2.Delete from the Rear\n3.Delete from the Front\n4.Display\n5.Exit\n";

while(1)

{

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:dq.insertRear();

break;

case 2:dq.deleteRear();

break;

case 3:dq.deleteFront();

break;

case 4:dq.fndisplay();

break;

case 5:exit(0);

}

}

}

OUTPUT:

MENU

1.Insert at the Rear

2.Delete from the Rear

3.Delete from the Front

4.Display

5.Exit

Enter your choice:1

Enter the element:1

Enter your choice:1

Enter the element:2

Enter your choice:1

Enter the element:3

Enter your choice:1

Enter the element:4

Enter your choice:1

Enter the element:5

Enter your choice:1

DeQueue is full!

Enter your choice:4

1 2 3 4 5

Enter your choice:2

Deleted element:5

Enter your choice:3

Deleted element:1

Enter your choice:4

2 3 4

Enter your choice:2

Deleted element:4

Enter your choice:3

Deleted element:2

Enter your choice:4

3

Enter your choice:2

Deleted element:3

Enter your choice:2

DeQueue is empty!

Enter your choice:4

Queue is empty!

Enter your choice:5

-----------------------------------------------------------------------------------------------------------------

9.Write a c++ program to implement Ascending Priority Queue using arrays and class.

#include<iostream>

#define MAX 5

using namespace std;

class PQueue

{

int front,rear;

int PQ[MAX];

public:

PQueue()

{

front=rear=-1;

}

void fninsert()

{

int obj;

if(rear==MAX-1)

cout<<"Queue is full!\n";

else

{

cout<<"Enter the element:";

cin>>obj;

PQ[++rear]=obj;

if(front==-1)

front=0;

}

}

void fndelete()

{

int obj,min,i,index=0;

if(front==-1)

cout<<"PQueue is empty!\n";

else if(front==rear)

front=rear=-1;

else

{

min=PQ[0];

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

{

if(min>PQ[i])

{

min=PQ[i];

index=i;

}

}

obj=PQ[index];

cout<<"Deleted element:"<<obj<<"\n";

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

{

PQ[i]=PQ[i+1];

}

rear=rear-1;

}

}

void fndisplay()

{

int i;

if(front==-1)

cout<<"Queue is empty!\n";

else

{

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

{

cout<<PQ[i]<<"\t";

}

}

}

};

int main()

{

int ch;

PQueue pq;

cout<<"MENU\n1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n";

while(1)

{

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:pq.fninsert();

break;

case 2:pq.fndelete();

break;

case 3:pq.fndisplay();

break;

case 4:exit(0);

}

}

}

OUTPUT:

MENU

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice:1

Enter the element:1

Enter your choice:1

Enter the element:6

Enter your choice:1

Enter the element:4

Enter your choice:1

Enter the element:9

Enter your choice:1

Enter the element:2

Enter your choice:1

Queue is full!

Enter your choice:3

1 6 4 9 2

Enter your choice:2

Deleted element:1

Enter your choice:2

Deleted element:2

Enter your choice:2

Deleted element:4

Enter your choice:2

Deleted element:6

Enter your choice:2

Enter your choice:2

PQueue is empty!

Enter your choice:3

Queue is empty!

Enter your choice:4

--------------------------------------------------------------------------------------------------------------

10.Write a c++ program to implement Singly Linked List using class and structure.

#include <iostream>

using namespace std;

typedef struct snode {

int data;

struct snode \*ptr;

} Node;

class Slist {

public:

Node \*fnInsertBegin(Node \*head) {

int num;

cout << "Enter data to insert at beginning: ";

cin >> num;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

newnode->ptr = head;

head = newnode;

return head;

}

Node \*fnInsertEnd(Node \*head) {

int num;

cout << "Enter the number to insert at the end: ";

cin >> num;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

newnode->ptr = NULL;

if (head == NULL) {

head = newnode;

} else {

Node \*temp = head;

while (temp->ptr != NULL)

temp = temp->ptr;

temp->ptr = newnode;

}

return head;

}

Node \*fnInsertpos(Node \*head) {

int num, pos;

cout << "Enter the number: ";

cin >> num;

cout << "Enter the position: ";

cin >> pos;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

if (pos == 0) {

newnode->ptr = head;

head = newnode;

} else {

Node \*temp = head;

int count = 0;

while (temp != NULL && count < pos - 1) {

temp = temp->ptr;

count++;

}

if (temp == NULL) {

cout << "Position out of bounds.\n";

free(newnode);

return head;

}

newnode->ptr = temp->ptr;

temp->ptr = newnode;

}

return head;

}

Node \*fnDeleteBegin(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

Node \*temp = head;

head = head->ptr;

cout<<"Deleted node from the beginning:"<<temp->data<<"\n";

free(temp);

return head;

}

Node \*fnDeleteEnd(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

if (head->ptr == NULL) {

cout<<"Deleted node from the end:"<<head->data<<"\n";

free(head);

return NULL;

}

Node \*temp = head;

while (temp->ptr->ptr != NULL)

temp = temp->ptr;

cout<<"Deleted node from the end:"<<temp->ptr->data<<"\n";

free(temp->ptr);

temp->ptr = NULL;

return head;

}

Node \*fnDeletepos(Node \*head, int pos) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

if (pos == 0) {

Node \*temp = head;

head = head->ptr;

cout<<"Deleted node from the position "<<pos<<":"<<temp->data<<"\n";

free(temp);

return head;

}

Node \*temp = head;

int count = 0;

while (temp->ptr != NULL && count < pos - 1) {

temp = temp->ptr;

count++;

}

if (temp->ptr == NULL) {

cout << "Position invalid.\n";

return head;

}

Node \*temp2 = temp->ptr;

temp->ptr = temp2->ptr;

cout<<"Deleted node from the position "<<pos<<":"<<temp2->data<<"\n";

free(temp2);

return head;

}

void fnDisplay(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return;

}

Node \*temp = head;

while (temp != NULL) {

cout << temp->data;

if (temp->ptr != NULL)

cout << " -> ";

temp = temp->ptr;

}

cout << "\n\*\*\* End \*\*\*\n";

}

};

int main() {

Slist s;

Node \*head = NULL;

int ch, pos;

cout << "Menu\n1. Insert begin\n2. Insert end\n3. Insert position\n4. Delete begin\n5. Delete end\n6. Delete position\n7. Display\n8. Exit\n";

while (1) {

cout << "Enter your choice: ";

cin >> ch;

switch (ch) {

case 1:

head = s.fnInsertBegin(head);

break;

case 2:

head = s.fnInsertEnd(head);

break;

case 3:

head = s.fnInsertpos(head);

break;

case 4:

head = s.fnDeleteBegin(head);

break;

case 5:

head = s.fnDeleteEnd(head);

break;

case 6:

cout << "Enter position to delete: ";

cin >> pos;

head = s.fnDeletepos(head, pos);

break;

case 7:

s.fnDisplay(head);

break;

case 8:

exit(0);

default:

cout << "Invalid choice. Try again.\n";

}

}

}

OUTPUT:

Menu

1. Insert begin

2. Insert end

3. Insert position

4. Delete begin

5. Delete end

6. Delete position

7. Display

8. Exit

Enter your choice: 1

Enter data to insert at beginning: 1

Enter your choice: 2

Enter the number to insert at the end: 2

Enter your choice: 1

Enter data to insert at beginning: 4

Enter your choice: 2

Enter the number to insert at the end: 6

Enter your choice: 3

Enter the number: 5

Enter the position: 2

Enter your choice: 7

4 -> 1 -> 5 -> 2 -> 6

\*\*\* End \*\*\*

Enter your choice: 4

Deleted node from the beginning:4

Enter your choice: 5

Deleted node from the end:6

Enter your choice: 7

1 -> 5 -> 2

\*\*\* End \*\*\*

Enter your choice: 6

Enter position to delete: 2

Deleted node from the position2:2

Enter your choice: 4

Deleted node from the beginning:1

Enter your choice: 5

Deleted node from the end:5

Enter your choice: 7

List is empty.

Enter your choice: 8

---------------------------------------------------------------------------------------------------------

11.Write a c++ program to implement Circular Linked List using class and structure.

#include <iostream>

using namespace std;

typedef struct cnode {

int data;

struct cnode \*ptr;

} Node;

class Clist {

public:

Node \*fnInsertBegin(Node \*head) {

int num;

cout << "Enter data to insert at beginning: ";

cin >> num;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

if (head == NULL) {

newnode->ptr = newnode; // Circular link

head = newnode;

} else {

Node \*temp = head;

while (temp->ptr != head)

temp = temp->ptr;

newnode->ptr = head;

temp->ptr = newnode;

head = newnode;

}

return head;

}

Node \*fnInsertEnd(Node \*head) {

int num;

cout << "Enter the number to insert at the end: ";

cin >> num;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

if (head == NULL) {

newnode->ptr = newnode; // Circular link

head = newnode;

} else {

Node \*temp = head;

while (temp->ptr != head)

temp = temp->ptr;

temp->ptr = newnode;

newnode->ptr = head; // Circular link

}

return head;

}

Node \*fnInsertpos(Node \*head) {

int num, pos;

cout << "Enter the number: ";

cin >> num;

cout << "Enter the position: ";

cin >> pos;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

if (head == NULL) {

if (pos == 0) {

newnode->ptr = newnode; // Circular link

head = newnode;

} else {

cout << "Invalid position for empty list.\n";

free(newnode);

}

} else if (pos == 0) {

Node \*temp = head;

while (temp->ptr != head)

temp = temp->ptr;

newnode->ptr = head;

temp->ptr = newnode;

head = newnode;

} else {

Node \*temp = head;

int count = 0;

while (temp->ptr != head && count < pos - 1) {

temp = temp->ptr;

count++;

}

if (count != pos - 1) {

cout << "Position out of bounds.\n";

free(newnode);

} else {

newnode->ptr = temp->ptr;

temp->ptr = newnode;

}

}

return head;

}

Node \*fnDeleteBegin(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

Node \*temp = head, \*last = head;

while (last->ptr != head)

last = last->ptr;

if (head->ptr == head) { // Only one node

cout << "Deleted element: " << head->data << endl;

free(head);

return NULL;

} else {

cout << "Deleted element: " << head->data << endl;

head = head->ptr;

last->ptr = head;

free(temp);

}

return head;

}

Node \*fnDeleteEnd(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

Node \*temp = head, \*prev = NULL;

while (temp->ptr != head) {

prev = temp;

temp = temp->ptr;

}

if (prev == NULL) { // Only one node

cout << "Deleted element: " << head->data << endl;

free(head);

return NULL;

} else {

cout << "Deleted element: " << temp->data << endl;

prev->ptr = head;

free(temp);

}

return head;

}

Node \*fnDeletepos(Node \*head, int pos) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

if (pos == 0) {

return fnDeleteBegin(head);

}

Node \*temp = head, \*prev = NULL;

int count = 0;

while (temp->ptr != head && count < pos) {

prev = temp;

temp = temp->ptr;

count++;

}

if (count != pos) {

cout << "Position invalid.\n";

} else {

cout << "Deleted element: " << temp->data << endl;

prev->ptr = temp->ptr;

free(temp);

}

return head;

}

void fnDisplay(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return;

}

Node \*temp = head;

do {

cout << temp->data << " -> ";

temp = temp->ptr;

} while (temp != head);

cout << "(back to head)\n";

}

};

int main() {

Clist c;

Node \*head = NULL;

int ch, pos;

cout << "Menu\n1. Insert begin\n2. Insert end\n3. Insert position\n4. Delete begin\n5. Delete end\n6. Delete position\n7. Display\n8. Exit\n";

while (1) {

cout << "Enter your choice: ";

cin >> ch;

switch (ch) {

case 1:

head = c.fnInsertBegin(head);

break;

case 2:

head = c.fnInsertEnd(head);

break;

case 3:

head = c.fnInsertpos(head);

break;

case 4:

head = c.fnDeleteBegin(head);

break;

case 5:

head = c.fnDeleteEnd(head);

break;

case 6:

cout << "Enter position to delete: ";

cin >> pos;

head = c.fnDeletepos(head, pos);

break;

case 7:

c.fnDisplay(head);

break;

case 8:

exit(0);

default:

cout << "Invalid choice. Try again.\n";

}

}

}

OUTPUT:

Menu

1. Insert begin

2. Insert end

3. Insert position

4. Delete begin

5. Delete end

6. Delete position

7. Display

8. Exit

Enter your choice: 1

Enter data to insert at beginning: 1

Enter your choice: 2

Enter the number to insert at the end: 2

Enter your choice: 3

Enter the number: 1

Enter the position: 1

Enter your choice: 1

Enter data to insert at beginning: 4

Enter your choice: 2

Enter the number to insert at the end: 7

Enter your choice: 7

4 -> 1 -> 1 -> 2 -> 7 -> (back to head)

Enter your choice: 4

Deleted element: 4

Enter your choice: 5

Deleted element: 7

Enter your choice: 7

1 -> 1 -> 2 -> (back to head)

Enter your choice: 6

Enter position to delete: 2

Deleted element: 2

Enter your choice: 7

1 -> 1 -> (back to head)

Enter your choice: 1

Enter data to insert at beginning: 6

Enter your choice: 2

Enter the number to insert at the end: 8

Enter your choice: 7

6 -> 1 -> 1 -> 8 -> (back to head)

Enter your choice: 8

---------------------------------------------------------------------------------------------------------

12.Write a c++ program to implement Doubly Linked List using class and structure.

#include <iostream>

using namespace std;

typedef struct dnode {

int data;

struct dnode \*next;

struct dnode \*prev;

} Node;

class Dlist {

public:

Node \*fnInsertBegin(Node \*head) {

int num;

cout << "Enter data to insert at beginning: ";

cin >> num;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

newnode->prev = NULL;

newnode->next = head;

if (head != NULL)

head->prev = newnode;

head = newnode;

return head;

}

Node \*fnInsertEnd(Node \*head) {

int num;

cout << "Enter the number to insert at the end: ";

cin >> num;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

newnode->next = NULL;

if (head == NULL) {

newnode->prev = NULL;

head = newnode;

} else {

Node \*temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newnode;

newnode->prev = temp;

}

return head;

}

Node \*fnInsertpos(Node \*head) {

int num, pos;

cout << "Enter the number: ";

cin >> num;

cout << "Enter the position: ";

cin >> pos;

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

if (pos == 0) {

newnode->prev = NULL;

newnode->next = head;

if (head != NULL)

head->prev = newnode;

head = newnode;

} else {

Node \*temp = head;

int count = 0;

while (temp != NULL && count < pos - 1) {

temp = temp->next;

count++;

}

if (temp == NULL) {

cout << "Position out of bounds.\n";

free(newnode);

} else {

newnode->next = temp->next;

newnode->prev = temp;

if (temp->next != NULL)

temp->next->prev = newnode;

temp->next = newnode;

}

}

return head;

}

Node \*fnDeleteBegin(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

Node \*temp = head;

cout << "Deleted element: " << temp->data << endl;

head = head->next;

if (head != NULL)

head->prev = NULL;

free(temp);

return head;

}

Node \*fnDeleteEnd(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

Node \*temp = head;

while (temp->next != NULL)

temp = temp->next;

cout << "Deleted element: " << temp->data << endl;

if (temp->prev != NULL)

temp->prev->next = NULL;

else

head = NULL;

free(temp);

return head;

}

Node \*fnDeletepos(Node \*head, int pos) {

if (head == NULL) {

cout << "List is empty.\n";

return head;

}

Node \*temp = head;

if (pos == 0) {

return fnDeleteBegin(head);

}

int count = 0;

while (temp != NULL && count < pos) {

temp = temp->next;

count++;

}

if (temp == NULL) {

cout << "Position invalid.\n";

} else {

cout << "Deleted element: " << temp->data << endl;

if (temp->prev != NULL)

temp->prev->next = temp->next;

if (temp->next != NULL)

temp->next->prev = temp->prev;

free(temp);

}

return head;

}

void fnDisplay(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return;

}

Node \*temp = head;

while (temp != NULL) {

cout << temp->data << " <-> ";

temp = temp->next;

}

cout << "NULL\n";

}

};

int main() {

Dlist d;

Node \*head = NULL;

int ch, pos;

cout << "Menu\n1. Insert begin\n2. Insert end\n3. Insert position\n4. Delete begin\n5. Delete end\n6. Delete position\n7. Display\n8. Exit\n";

while (1) {

cout << "Enter your choice: ";

cin >> ch;

switch (ch) {

case 1:

head = d.fnInsertBegin(head);

break;

case 2:

head = d.fnInsertEnd(head);

break;

case 3:

head = d.fnInsertpos(head);

break;

case 4:

head = d.fnDeleteBegin(head);

break;

case 5:

head = d.fnDeleteEnd(head);

break;

case 6:

cout << "Enter position to delete: ";

cin >> pos;

head = d.fnDeletepos(head, pos);

break;

case 7:

d.fnDisplay(head);

break;

case 8:

exit(0);

default:

cout << "Invalid choice. Try again.\n";

}

}

}

OUTPUT:

Menu

1. Insert begin

2. Insert end

3. Insert position

4. Delete begin

5. Delete end

6. Delete position

7. Display

8. Exit

Enter your choice: 1

Enter data to insert at beginning: 1

Enter your choice: 2

Enter the number to insert at the end: 2

Enter your choice: 1

Enter data to insert at beginning: 5

Enter your choice: 2

Enter the number to insert at the end: 8

Enter your choice: 3

Enter the number: 4

Enter the position: 3

Enter your choice: 7

5 <-> 1 <-> 2 <-> 4 <-> 8 <-> NULL

Enter your choice: 4

Deleted element: 5

Enter your choice: 5

Deleted element: 8

Enter your choice: 7

1 <-> 2 <-> 4 <-> NULL

Enter your choice: 6

Enter position to delete: 1

Deleted element: 2

Enter your choice: 7

1 <-> 4 <-> NULL

Enter your choice: 1

Enter data to insert at beginning: 3

Enter your choice: 7

3 <-> 1 <-> 4 <-> NULL

Enter your choice: 8

------------------------------------------------------------------------------------------------------------

13.Write a c++ program to implement Ordered Singly Linked List,Reverse the list and display the same using class and structure.

#include <iostream>

using namespace std;

typedef struct snode {

int data;

struct snode \*next;

} Node;

class OrderedSlist {

public:

Node \*fnInsertOrdered(Node \*head, int num) {

Node \*newnode = (Node \*)malloc(sizeof(Node));

newnode->data = num;

newnode->next = NULL;

if (head == NULL || head->data >= num) {

// Insert at the beginning

newnode->next = head;

head = newnode;

} else {

// Insert in the middle or at the end

Node \*temp = head;

while (temp->next != NULL && temp->next->data < num) {

temp = temp->next;

}

newnode->next = temp->next;

temp->next = newnode;

}

return head;

}

Node \*fnReverse(Node \*head) {

Node \*prev = NULL, \*current = head, \*next = NULL;

while (current != NULL) {

next = current->next; // Store the next node

current->next = prev; // Reverse the link

prev = current; // Move prev one step forward

current = next; // Move current one step forward

}

head = prev; // Update head to the last non-null node

return head;

}

void fnDisplay(Node \*head) {

if (head == NULL) {

cout << "List is empty.\n";

return;

}

Node \*temp = head;

while (temp != NULL) {

cout << temp->data << " -> ";

temp = temp->next;

}

cout << "NULL\n";

}

};

int main() {

OrderedSlist oslist;

Node \*head = NULL;

int ch, num;

cout << "Menu:\n";

cout << "1. Insert ordered\n";

cout << "2. Display\n";

cout << "3. Reverse and display\n";

cout << "4. Exit\n";

while (1) {

cout << "Enter your choice: ";

cin >> ch;

switch (ch) {

case 1:

cout << "Enter a number to insert: ";

cin >> num;

head = oslist.fnInsertOrdered(head, num);

break;

case 2:

cout << "List: ";

oslist.fnDisplay(head);

break;

case 3:

head = oslist.fnReverse(head);

cout << "Reversed list: ";

oslist.fnDisplay(head);

break;

case 4:

exit(0);

default:

cout << "Invalid choice. Try again.\n";

}

}

return 0;

}

OUTPUT:

Menu:

1. Insert ordered

2. Display

3. Reverse and display

4. Exit

Enter your choice: 1

Enter a number to insert: 4

Enter your choice: 1

Enter a number to insert: 2

Enter your choice: 1

Enter a number to insert: 7

Enter your choice: 1

Enter a number to insert: 5

Enter your choice: 2

List: 2 -> 4 -> 5 -> 7 -> NULL

Enter your choice: 3

Reversed list: 7 -> 5 -> 4 -> 2 -> NULL

Enter your choice: 4

-------------------------------------------------------------------------------------------------------------------

14.Write a c++ program to implement binary tree using linked list and traverse the tree using inorder,pre order and post order traversal methods.

#include <iostream>

using namespace std;

typedef struct btnode{

int data;

struct btnode \*left;

struct btnode \*right;

} Node;

class BTList{

public:

Node \*insert(Node \*);

void inorder(Node \*);

void preorder(Node \*);

void postorder(Node \*);

};

int main(){

int ch, level = 1;

BTList bt;

Node \*root = NULL;

cout<<"\nMenu\n1.Insert\n2.Inorder\n3.Preorder\n4.Postorder\n5.Exit\n"<<endl;

while (true){

cout<<"Enter your choice : ";

cin>>ch;

switch (ch){

case 1:

root = bt.insert(root);

break;

case 2:

cout<<"In-Order Traversal :";

bt.inorder(root);

cout<<"\n";

break;

case 3:

cout<<"Pre-Order Traversal:";

bt.preorder(root);

cout<<"\n";

break;

case 4:

cout<<"Post-Order Traversal:";

bt.postorder(root);

cout<<"\n";

break;

case 5:

exit(0);

default:

cout<<"Invalid Choice!";

break;

}

}

return 0;

}

Node \*BTList::insert(Node \*root){

int item, num;

Node \*newNode, \*cur, \*ptr;

newNode = (Node \*)malloc(sizeof(Node));

cout<<"Enter the element to insert : ";

cin>>num;

newNode -> data = num;

newNode -> left = NULL;

newNode -> right = NULL;

item = newNode -> data;

if (root == NULL)

root = newNode;

else{

cur = root;

while (cur != NULL){

ptr = cur;

if (item >= cur -> data)

cur = cur -> right;

else

cur = cur -> left;

}

if (item >= ptr -> data)

ptr -> right = newNode;

else

ptr -> left = newNode;

}

return root;

}

void BTList ::inorder(Node \*root){

if (root == NULL) return;

inorder(root -> left);

cout<<root -> data<<" ";

inorder(root -> right);

}

void BTList ::preorder(Node \*root){

if (root == NULL) return;

cout<<root -> data<<" ";

preorder(root -> left);

preorder(root -> right);

}

void BTList ::postorder(Node \*root){

if (root == NULL) return;

postorder(root -> left);

postorder(root -> right);

cout<<root -> data<<" ";

}

OUTPUT:

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Exit

Enter your choice : 1

Enter the element to insert : 4

Enter your choice : 1

Enter the element to insert : 5

Enter your choice : 1

Enter the element to insert : 9

Enter your choice : 1

Enter the element to insert : 7

Enter your choice : 1

Enter the element to insert : 8

Enter your choice : 1

Enter the element to insert : 11

Enter your choice : 1

Enter the element to insert : 3

Enter your choice : 2

In-Order Traversal :3 4 5 7 8 9 11

Enter your choice : 3

Pre-Order Traversal:4 3 5 9 7 8 11

Enter your choice : 4

Post-Order Traversal:3 8 7 11 9 5 4

Enter your choice : 5

-------------------------------------------------------------------------------------------------------------------------------

15.Write a c++ program to implement binary tree using array and display the content of the array.

#include <iostream>

using namespace std;

class BinaryTree {

public:

int\* tree; // Array to store binary tree

int size; // Size of the binary tree

// Constructor to initialize the binary tree

BinaryTree(int n) {

size = n;

tree = new int[size]; // Allocate memory for the array

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

tree[i] = -1; // Initialize all elements as -1 (empty)

}

}

// Insert a value at a specific index

void insert(int index, int value) {

if (index < size) {

tree[index] = value; // Insert value at the specified index

}

}

// Display the contents of the binary tree

void display() {

cout << "Binary Tree: ";

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

if (tree[i] != -1) {

cout << tree[i] << " ";

}

}

cout << endl;

}

};

int main() {

int n;

cout << "Enter the number of nodes in the binary tree: ";

cin >> n;

BinaryTree bt(n); // Create binary tree with size 'n'

cout << "Enter the elements for the binary tree:\n";

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

int value;

cin >> value;

bt.insert(i, value); // Insert value at index 'i'

}

// Display the binary tree

bt.display();

return 0;

}

OUTPUT:

Enter the number of nodes in the binary tree: 7

Enter the elements for the binary tree:

4

5

9

7

8

11

3

Binary Tree: 4 5 9 7 8 11 3

--------------------------------------------------------------------------------------------------------

16.Write a c++ proram to sort the array elements using Bubble sort.

#include <iostream>

using namespace std;

void bubbleSort(int[], int);

int main(){

int x[10], n;

cout<<"Enter no of Elements : ";

cin>>n;

cout<<"Enter Elements in array :\n";

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

cin>>x[i];

bubbleSort(x, n);

return 0;

}

void bubbleSort(int x[10], int n){

int i, j, temp;

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

for(j = 0; j<n-i-1; j++){

if(x[j] > x[j+1]) {

temp = x[j];

x[j] = x[j+1];

x[j+1] = temp;

}

}

}

cout<<"\nSorted Elements using Bubble sort : ";

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

cout<<x[i]<<" ";

}

OUTPUT:

Enter no of Elements : 5

Enter Elements in array :

7

4

2

8

1

Sorted Elements using Bubble sort : 1 2 4 7 8

---------------------------------------------------------------------------------------------

17.Write a c++ proram to sort the array elements using Insertion sort.

#include <iostream>

using namespace std;

void insertionSort(int[], int);

int main(){

int x[10], n;

cout<<"Enter no of Elements: ";

cin>>n;

cout<<"Enter Integers in an Array:\n";

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

cin>>x[i];

insertionSort(x, n);

return 0;

}

void insertionSort(int x[10], int n){

int i, j, key;

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

key = x[i];

for(j=i-1; j>=0 && key<x[j]; j--)

{

x[j+1] = x[j];

}

x[j+1] = key;

}

cout<<"Sorted Array using Insertion Sort : ";

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

cout<<x[i]<<" ";

}

OUTPUT:

Enter no of Elements: 5

Enter Integers in an Array:

6

3

5

1

2

Sorted Array using Insertion Sort : 1 2 3 5 6

----------------------------------------------------------------------------------------------------------

18.Write a c++ proram to sort the array elements using Selection sort.

#include <iostream>

using namespace std;

void selectionSort(int [], int);

int main(){

int x[10], n, i;

cout<<"Enter number of Elements : ";

cin>>n;

cout<<"Enter Elements :\n";

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

cin>>x[i];

selectionSort(x, n);

return 0;

}

void selectionSort(int x[10], int n){

int i, j, large, index;

for(i = n-1; i>0; i--){

large = x[0];

index = 0;

for(j = 1; j<=i; j++){

if(x[j]>large) {

large = x[j];

index = j;

}

}

x[index] = x[i];

x[i] = large;

}

cout<<"Sorted Array using Selection sort : ";

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

cout<<x[i]<<" ";

}

OUTPUT:

Enter number of Elements : 5

Enter Elements :

8

3

9

2

4

Sorted Array using Selection sort : 2 3 4 8 9

----------------------------------------------------------------------------------------------------------

19.Write a c++ program to sort the array elements using Quick sort.

#include <iostream>

using namespace std;

int quick(int a[], int low, int high);

int partition(int a[], int low, int high);

int main()

{

int i, n, j, a[20], low, high;

cout << "Enter the size of the array: ";

cin >> n;

cout << "Enter the elements:\n";

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

cin >> a[i];

low = 0;

high = n - 1;

quick(a, low, high);

cout << "Sorted Array using Quick Sort: ";

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

{

cout << a[i] << " ";

}

cout << endl;

return 0;

}

int quick(int a[], int low, int high)

{

int keypos;

if (low >= high)

return 0;

else

{

keypos = partition(a, low, high);

quick(a, low, keypos - 1);

quick(a, keypos + 1, high);

}

return 0;

}

int partition(int a[], int low, int high)

{

int i, j, key, temp;

i = low + 1;

j = high;

key = a[low];

while (1)

{

while (key > a[i] && i < high)

i++;

while (key < a[j])

j--;

if (i < j)

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

else

{

temp = a[low];

a[low] = a[j];

a[j] = temp;

return j;

}

}

}

OUTPUT:

Enter the size of the array: 5

Enter the elements:

8

1

5

2

9

Sorted Array using Quick Sort: 1 2 5 8 9

----------------------------------------------------------------------------------------------------------

20.Write a c++ program to sort the array elements using Merge sort.

#include <iostream>

using namespace std;

void merge(int a[], int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int leftArr[n1], rightArr[n2];

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

leftArr[i] = a[left + i];

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

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

int i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

if (leftArr[i] <= rightArr[j])

a[k++] = leftArr[i++];

else

a[k++] = rightArr[j++];

}

while (i < n1)

a[k++] = leftArr[i++];

while (j < n2)

a[k++] = rightArr[j++];

}

void mergeSort(int a[], int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(a, left, mid);

mergeSort(a, mid + 1, right);

merge(a, left, mid, right);

}

}

int main() {

int a[10], n;

cout << "Enter the size of the array: ";

cin >> n;

cout << "Enter array elements:\n";

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

cin >> a[i];

mergeSort(a, 0, n - 1);

cout << "Sorted array:\n";

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

cout << a[i] << " ";

return 0;

}

OUTPUT:

Enter number of Elements : 5

Enter the Elements :

7

2

9

11

3

Sorted Elements using Merge sort:2 3 7 9 11

----------------------------------------------------------------------------------------------------------

21.Write a c++ program to sort the array elements using Heap sort.

#include<iostream>

using namespace std;

void heapify(int arr[],int n,int i)

{

int largest=i;

int left=2\*i+1;

int right=2\*i+2;

if(left<n&&arr[left]>arr[largest])

largest=left;

if(right<n&&arr[right]>arr[largest])

largest=right;

if(largest!=i)

{

swap(arr[i],arr[largest]);

heapify(arr,n,largest);

}

}

void heapSort(int a[],int n)

{

for(int i=n/2-1;i>=0;i--)

{

heapify(a,n,i);

}

for(int i=n-1;i>0;i--)

{

swap(a[0],a[i]);

heapify(a,i,0);

}

}

void printArray(int a[],int n)

{

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

{

cout<<a[i]<<" ";

}

cout<<"\n";

}

int main()

{

int a[10],n,i;

cout<<"Enter the no.of elements:";

cin>>n;

cout<<"Enter the array elements:\n";

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

{

cin>>a[i];

}

cout<<"Original array:";

printArray(a,n);

heapSort(a,n);

cout<<"After sorting:";

printArray(a,n);

return 0;

}

OUTPUT:

Enter the no.of elements:5

Enter the array elements:

7

1

9

3

2

Original array:7 1 9 3 2

After sorting:1 2 3 7 9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*