**PRACTICAL NO – 01**

**Implement the following:**

**A) Write a program to store the elements in 1-D array and perform the operations like searching, sorting and reversing the elements. [Menu Driven]**

**Code :**

#include <iostream>

using namespace std;

intarr[20] = {} , i , j , chc , size , t;

intarr\_print()

{

cout<<"Array : ";

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

{

cout<<arr[i]<<" ";

}

}

intarr\_insert()

{

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

cin>>size;

arr[size];

cout<<endl;

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

{

cout<<"Enter element "<<i + 1<<" : ";

cin>>arr[i];

}

system("cls");

}

intarr\_search()

{

int f = 0;

cout<<"Enter the Element to Search : ";

cin>>chc;

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

{

if(arr[i] == chc)

{

f = 1;

break;

}

}

if(f == 1)

{

cout<<"Element Found...\n";

}

else

{

cout<<"Element not Found...\n";

}

}

intarr\_sort()

{

cout<<"1 ] Ascending Sort\n2 ] Descending Sort\nChoice>>>>> ";

cin>>chc;

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

{

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

{

if(chc == 1)

{

if(arr[i] >arr[j])

{

t = arr[i];

arr[i] = arr[j];

arr[j] = t;

}

}

else if(chc == 2)

{

if(arr[i] <arr[j])

{

t = arr[i];

arr[i] = arr[j];

arr[j] = t;

}

}

}

}

arr\_print();

}

intarr\_reverse()

{

i = 0;

chc = size;

chc -= 1;

while(i<chc)

{

t = arr[i];

arr[i] = arr[chc];

arr[chc] = t;

i++;

chc--;

}

arr\_print();

}

int menu()

{

intyn;

do

{

system("cls");

cout<<"Array Operations :\n1 ] Search\n2 ] Sort\n3 ] Reverse\n4 ] Exit\nChoice>>>> ";

cin>>chc;

switch(chc)

{

case 1 :

system("cls");

arr\_search();

break;

case 2 :

system("cls");

arr\_sort();

break;

case 3 :

system("cls");

arr\_reverse();

break;

case 4 :

exit(0);

break;

default :

cout<<"Chooce a correct option...";

}

cout<<"\nDo you want to continue ?\n1 | YES 2 | NO\nChoice>>>>> ";

cin>>yn;

}while(yn != 2);

}

int main()

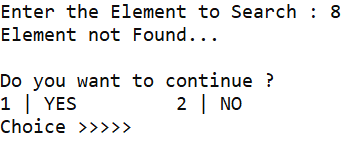
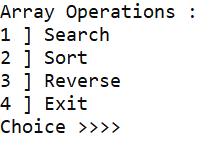
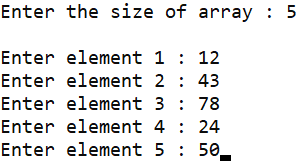
{

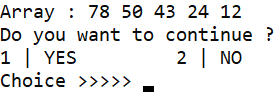
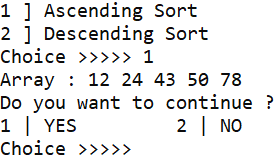
arr\_insert();

menu();

}

**Output :**

****

****

**B) Read the two arrays from the user and merge them and display the elements in sorted order. [Menu Driven]**

**Code :**

#include <iostream>

using namespace std;

inti , j , size1 , size2 , arr\_1[20] = {} , arr\_2[20] = {};

intarr\_print(intarr[] , int size)

{

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

{

cout<<arr[i]<<" ";

}

}

intarr\_insert()

{

cout<<"Enter the size of array 1 : ";

cin>>size1;

arr\_1[size1];

cout<<"Enter the size of array 2 : ";

cin>>size2;

arr\_2[size2];

cout<<endl;

cout<<"Array 1 elements :\n";

for(i = 0 ; i< size1 ; i++)

{

cout<<"Enter element "<<i + 1<<" : ";

cin>>arr\_1[i];

}

cout<<endl;

cout<<"Array 2 elements :\n";

for(i = 0 ; i< size2 ; i++)

{

cout<<"Enter element "<<i + 1<<" : ";

cin>>arr\_2[i];

}

system("cls");

cout<<"You have entered :\n\nArray 1 : ";

arr\_print(arr\_1 , size1);

cout<<"\nArray 2 : ";

arr\_print(arr\_2 , size2);

}

intarr\_sort(intarr[] , int size)

{

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

{

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

{

if(arr[i] >arr[j])

{

int t = arr[i];

arr[i] = arr[j];

arr[j] = t;

}

}

}

cout<<"\nMerged& Sorted Array : \n";

arr\_print(arr , size);

}

intarr\_merge()

{

int size = size1 + size2;

int merged[size];

j = 0;

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

{

if(i< size1)

{

merged[i] = arr\_1[i];

}

else

{

merged[i] = arr\_2[j];

j++;

}

}

arr\_sort(merged , size);

}

int main()

{

intchc;

do

{

system("cls");

arr\_insert();

arr\_merge();

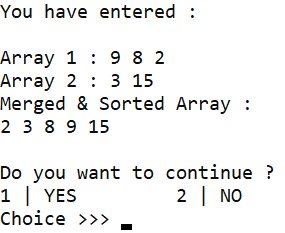
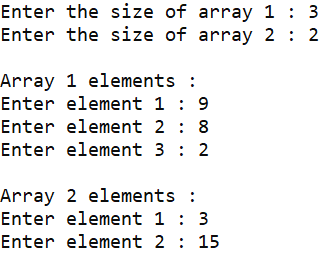
cout<<"\n\nDo you want to continue ?\n1 | YES 2 | NO\nChoice>>> ";

cin>>chc;

}while(chc != 2);

}

**Output :**

****

**C) Write a program to perform the Matrix addition, Multiplication and Transpose Operation. [Menu Driven]**

**Code :**

#include <iostream>

using namespace std;

int matrix\_1[20][20] = {} , matrix\_2[20][20] = {} , res[20][20] , i , j , chc , row1 , col1 , row2 , col2;

intmatrix\_print(int matrix[20][20] , int row , int col)

{

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

{

for(j = 0 ; j < col ; j++)

{

cout<<matrix[i][j]<<" ";

}

cout<<endl;

}

}

intmatrix\_insert(intcnt)

{

if(cnt == 2)

{

cout<<"Enter the row & column of Matrix 1 : ";

cin>>row1>>col1;

matrix\_1[row1][col1];

cout<<"Enter the row & column of Matrix 2 : ";

cin>>row2>>col2;

matrix\_2[row2][col2];

cout<<endl;

cout<<"Matrix 1 : \n";

for(i = 0 ; i< row1 ; i++)

{

for(j = 0 ; j < col1 ; j++)

{

cout<<"Element ["<<i + 1<<"]["<<j + 1<<"] : ";

cin>>matrix\_1[i][j];

}

}

cout<<endl;

cout<<"Matrix 2 : \n";

for(i = 0 ; i< row2 ; i++)

{

for(j = 0 ; j < col2 ; j++)

{

cout<<"Element ["<<i + 1<<"]["<<j + 1<<"] : ";

cin>>matrix\_2[i][j];

}

}

}

else

{

cout<<"Enter the row & column of Matrix : ";

cin>>row1>>col1;

matrix\_1[row1][col1];

cout<<endl;

cout<<"Matrix : \n";

for(i = 0 ; i< row1 ; i++)

{

for(j = 0 ; j < col1 ; j++)

{

cout<<"Element ["<<i + 1<<"]["<<j + 1<<"] : ";

cin>>matrix\_1[i][j];

}

}

}

}

intmatrix\_add()

{

if(row1 != row2 || col1 != col2)

{

cout<<"\nMatrix addition is not possible...\n";

}

else

{

for(i = 0 ; i< row1 ; i++)

{

for(j = 0 ; j < col1 ; j++)

{

res[i][j] = 0;

res[i][j] = matrix\_1[i][j] + matrix\_2[i][j];

}

}

cout<<"\nAddition of matrix is : \n";

matrix\_print(res , row1 , col1);

}

}

intmatrix\_mul()

{

for(i = 0 ; i< row1 ; i++)

{

for(j = 0 ; j < col2 ; j++)

{

res[i][j] = 0;

for(int k = 0 ; k < col1 ; k++)

{

res[i][j] += matrix\_1[i][k] \* matrix\_2[k][j];

}

}

}

matrix\_print(res , row1 , col2);

}

intmatrix\_trans()

{

res[col1][row1];

for(i = 0 ; i< row1 ; i++)

{

for(j = 0 ; j < col1 ; j++)

{

res[j][i] = matrix\_1[i][i];

}

}

matrix\_print(res , col1 , row1);

}

int menu()

{

intyn;

do

{

system("cls");

cout<<"Matrix Operations :\n1 ] Addition\n2 ] Multiplication\n3 ] Transpose\n4 ] Exit\nChoice>>>> ";

cin>>chc;

switch(chc)

{

case 1 :

system("cls");

matrix\_insert(2);

matrix\_add();

break;

case 2 :

system("cls");

matrix\_insert(2);

matrix\_mul();

break;

case 3 :

system("cls");

matrix\_insert(1);

matrix\_trans();

break;

case 4 :

exit(0);

break;

default :

cout<<"Chooce a correct option...";

}

cout<<"\nDo you want to continue ?\n1 | YES 2 | NO\nChoice>>>>> ";

cin>>yn;

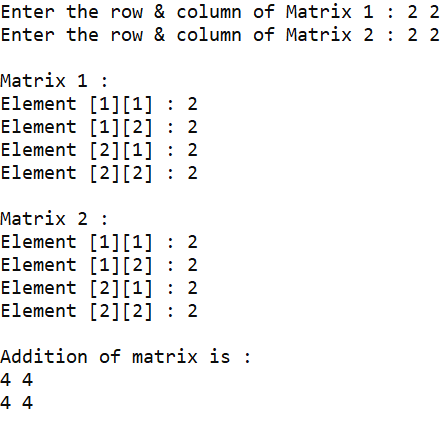
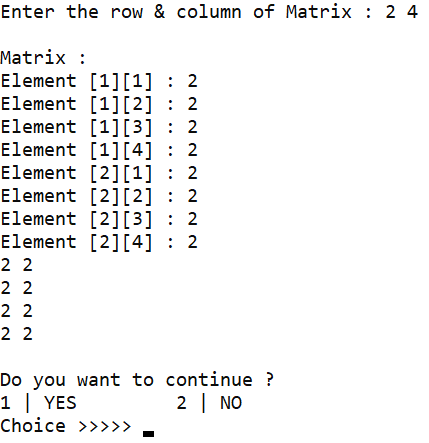
}while(yn != 2);

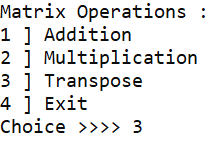
}

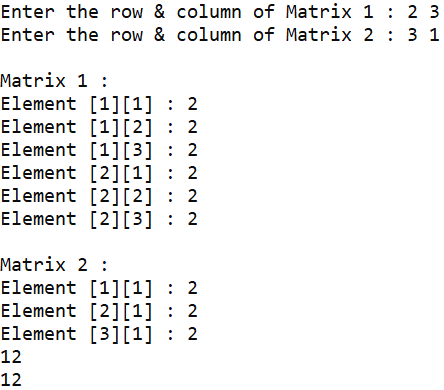
int main()

{

menu();

}****

**Output :**

****

**PRACTICAL NO – 02**

**Implement the following for Linked List**

**A) Write a program to create a single linked list and display the node elements in reverse order**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

intval , loop = 1 , size = 0;

struct node{

int data;

struct node \*next;

};

struct node \*head = NULL;

struct node \*temp , \*temp2;

voidcreate\_node(intval){

struct node \*new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = val;

new\_node -> next = NULL;

if(head == NULL){

head = new\_node;

temp = head;

}else{

temp -> next = new\_node;

temp = temp -> next;

temp -> next = NULL;

}

size++;

}

voidprint\_list(){

system("cls");

temp = head;

while(temp != NULL){

cout<<"\nData : "<<temp -> data<<"\nAddress : "<<temp<<endl;

temp = temp -> next;

}

}

voidreverse\_list(){

temp = NULL;

temp2 = NULL;

while(head != NULL){

temp2 = head -> next;

head -> next = temp;

temp = head;

head = temp2;

}

head = temp;

}

int main(){

inti = 0;

do{

i++;

system("cls");

cout<<"Element "<<i<<" : ";

cin>>val;

create\_node(val);

cout<<"Creation Successfull\n\nDo you want to continue ?\n[ 1 ] Yes No [ 2 ]\n\nChoice>>> ";

cin>>loop;

}while(loop != 2);

system("cls");

cout<<"\nLinkList : \n";

print\_list();

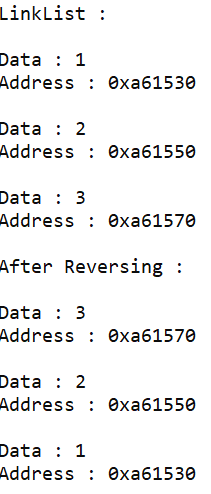
cout<<"\nAfter Reversing : \n";

reverse\_list();

print\_list();

}

**Output :**

****

**B) Write a program to search the elements in the linked list and display the same**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

inti , j , chc , size = 0 , val , loop = 1;

struct node{

int data;

struct node \*next;

};

struct node \*head = NULL;

struct node \*temp;

voidcreate\_node(intval){

struct node \*new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = val;

if(head == NULL){

head = new\_node;

head -> next = head;

temp = head;

}else{

new\_node -> next = head;

temp -> next = new\_node;

temp = new\_node;

}

size++;

}

voidsearch\_element(){

int value , f = 0;

cout<<"\nValue to search : ";

cin>>value;

temp = head;

do{

if(value == temp -> data){

f = 1;

break;

}

temp = temp -> next;

}while(temp -> next != head);

if(f){

cout<<"\nValue found and stored in "<<temp<<endl;

}else{

cout<<"\nValue not found and in link list";10

}

}

voidprint\_list(){

system("cls");

cout<<"\nCircular Link List :\n";

temp = head;

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

cout<<"\nData "<<i<<" : "<<temp -> data<<"\nAddress : "<<temp<<endl;

temp = temp -> next;

}

}

int main(){

i = 0;

do{

i++;

system("cls");

cout<<"Element "<<i<<" : ";

cin>>val;

create\_node(val);

cout<<"Creation Successfull\n\nDo you want to continue ?\n[ 1 ] Yes No [ 2 ]\n\nChoice>>> ";

cin>>loop;

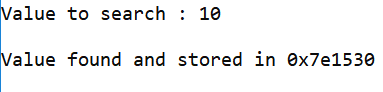
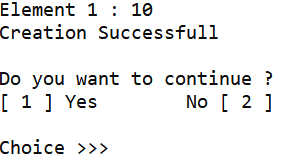
}while(loop != 2);

system("cls");

search\_element();

}

**Output :**

****

**C) Write a program to create double linked list and sort the elements in the linked list**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

inti , j , size = 0 , select ;

struct node{

int data;

struct node \*next , \*previous;

};

struct node \*head = NULL , \*temp , \*temp2;

voidprint\_list();

voiduser\_continue();

void menu();

voidisEmptyList(){

system("cls");

if(size == 0){

cout<<"\nDouble Link List is empty\n\n--------------------------------------\nPress any key ";

getch();

menu();

}

}

voidcreate\_node(intval){

struct node \*new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = val;

new\_node -> next = NULL;

new\_node -> previous = NULL;

if(head == NULL){

head = new\_node;

temp = head;

}else{

new\_node -> previous = temp;

temp -> next = new\_node;

temp = temp -> next;

}

size++;

}

voidsort\_list(){

system("cls");

int t;

temp = head;

while(temp != NULL){

temp2 = temp -> next;

while(temp2 != NULL){

if(temp -> data > temp2 -> data){

t = temp -> data;

temp -> data = temp2 -> data;

temp2 -> data = t;

}

temp2 = temp2 -> next;

}

temp = temp -> next;

}

cout<<"\nDouble Link List sorted successfully...\n";

}

void menu(){

int loop , i = 0;

system("cls");

cout<<"\n---------- Double Link List ----------\n\n[ 1 ] Create Link List\n[ 2 ] Display\n[ 3 ] Sort\n[ 4 ] Exit\n\n--------------------------------------\nChoice>>> ";

cin>>select;

switch(select){

case 1 :

do{

i++;

system("cls");

cout<<"Element "<<i<<" : ";

cin>>val;

create\_node(val);

cout<<"Creation Successfull\n\n--------------------------------------\nDo you want to continue ?\n[ 1 ] Yes No [ 2 ]\n\nChoice>>> ";

cin>>loop;

}while(loop != 2);

menu();

break;

case 2 :

print\_list();

user\_continue();

break;

case 3 :

sort\_list();

user\_continue();

break;

case 4 :

exit(0);

break;

}

}

voidprint\_list(){

isEmptyList();

cout<<"\nDouble Link List :\n";

temp = head;

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

cout<<"\nData "<<i<<" : "<<temp -> data<<"\nAddress : "<<temp<<endl;

temp = temp -> next;

}

}

voiduser\_continue(){

cout<<"\n--------------------------------------\nNavigate to ?\n[ 1 ] Menu Exit [ 2 ]\n\nChoice>>> ";

cin>>select;

if(select == 1){

menu();

}else{

exit(0);

}

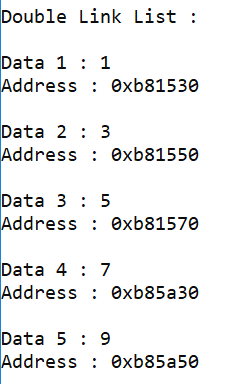
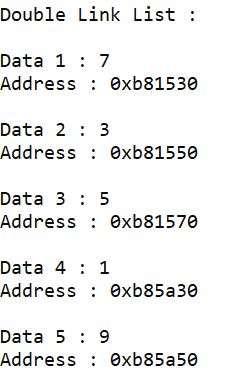
}

int main(){

menu();

}

**Output :**

****

**PRACTICAL NO – 03**

**Implement the following for Stack**

**A) Write a program to implement the concept of Stack with Push, Pop, Display and Exit operations**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

struct stack{

int data;

struct stack \*next;

};

int count = 0 , i = 1 , size = 5 , loop , chc , val;

struct stack \*top = NULL , \*temp;

voidisContinue();

void menu();

void push(intval){

if(count < size){

struct stack \*element = (struct stack \*)malloc(sizeof(struct stack));

element -> data = val;

element -> next = NULL;

if(top == NULL){

top = element;

temp = top;

}else{

element -> next = top;

top = element;

}

count++;

cout<<"Element pushed successfully!!\n";

}else{

cout<<"\nStack is in Overflow State!!\n";

}

}

void display(){

system("cls");

temp = top;

if(top == NULL){

cout<<"\nStack is in Underflow State!!\n";

}else{

cout<<"\nAvailable Stack :\n";

while(temp != NULL){

cout<<temp -> data<<endl;

temp = temp -> next;

}

}

isContinue();

}

void pop(){

system("cls");

if(top == NULL){

cout<<"\nStack is in Underflow State!!\n";

}else{

temp = top;

top = top -> next;

free(temp);

cout<<"\nElement popped successfully!!\n";

}

isContinue();

}

voidisContinue(){

cout<<"\nDo you want to continue ? \n[1 | MENU]\t\t\t[2 | EXIT]\n\nChoice>>> ";

cin>>loop;

if(loop){

menu();

}else{

exit(0);

}

}

void menu(){

system("cls");

cout<<"\n<<<<<<<<<<<<<<<<<<<<<<<<<< Stack Operations >>>>>>>>>>>>>>>>>>>>>>>>>>\n\n1 ] Push\n2 ] Display\n3 ] Pop\n4 ] Exit\n\nChoice>>> ";

cin>>chc;

switch(chc){

case 1 :

do{

system("cls");

cout<<"Value "<<i<<" : ";

cin>>val;

push(val);

cout<<"\nDo you want to continue ? \n[1 | YES]\t\t\t[2 | NO]\n\nChoice>>> ";

cin>>loop;

i++;

}while(loop != 2);

menu();

break;

case 2 :

display();

break;

case 3 :

pop();

break;

case 4 :

exit(0);

}

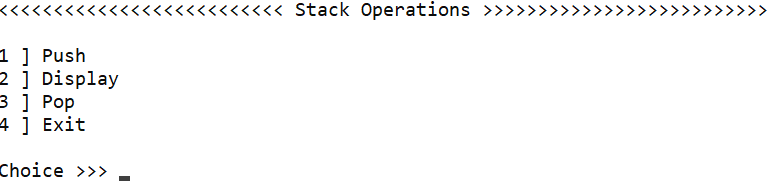
}

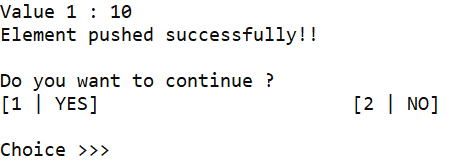
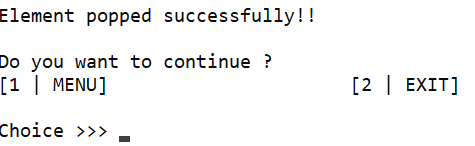
int main(){

menu();

}

**Output :**

****

****

**B) Write a program to convert an infix expression to postfix and prefix conversion**

**Code :**

#include <iostream>

#include <stack>

#include <algorithm>

using namespace std;

stringexp , exp1 , op = "";

inti , loop = 1;

intpriorityCheck(char op){

if(op == '('){

return 0;

}else if(op == '+' || op == '-'){

return 1;

}else if(op == '\*' || op == '/'){

return 2;

}else if(op == '^'){

return 3;

}

}

voidPostFixAndInFix(intval){

stack<char> stack;

op = "";

stack.push('(');

if(val == 1){

exp1 = exp;

reverse(exp1.begin() , exp1.end());

}

for(i = 0 ; i<exp.length() ; i++){

if(exp[i] == '('){

stack.push(exp[i]);

}else if(exp[i] == ')'){

while(stack.top() != '('){

op += stack.top();

stack.pop();

}

stack.pop();

}else if(exp[i] == '+' || exp[i] == '-' || exp[i] == '\*' || exp[i] == '/' || exp[i] == '^'){

while(priorityCheck(stack.top()) >= priorityCheck(exp[i]) && !stack.empty()){

op += stack.top();

stack.pop();

}

stack.push(exp[i]);

}else{

op += exp[i];

}

}

while(stack.top() != '('){

op += stack.top();

stack.pop();

}

if(val == 1){

reverse(op.begin(), op.end());

}

}

void menu(){

do{

system("cls");

cout<<"\nExpression : ";

cin>>exp;

cout<<"\n1 ] PostFix\n2 ] Prefix\n\nChoice>>> ";

cin>>i;

system("cls");

cout<<"\nInFix Expression : "<<exp;

if(i == 1){

PostFixAndInFix(0);

cout<<"\nPostFix Expression : "<<op<<endl;

}else if(i == 2){

PostFixAndInFix(1);

cout<<"\nPreFix Expression : "<<op<<endl;

}

cout<<"\n[1 | Home]\t\t[2 | Exit]\n\nChoice>>> ";

cin>>loop;

if(loop == 1){

menu();

}else if(loop == 2){

exit(0);

}

}while(loop != 2);

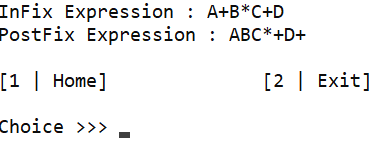
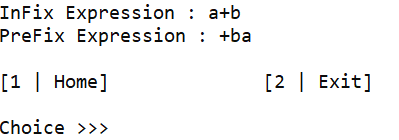
}

int main(){

menu();

}

**Output :**

****

**C) Write a program to implement Tower of Hanoi problem.**

**Code :**

#include<iostream>

using namespace std;

void toh(int n , char s , char a , char d){

if(n == 1){

cout<<"\nMove disk "<< n <<" from "<<s<<" to "<<d<<endl;

}else{

toh(n-1 , s , d , a);

cout<<"\nMove disk "<< n <<" from "<<s<<" to "<<d<<endl;

toh(n-1 , a , s , d);

}

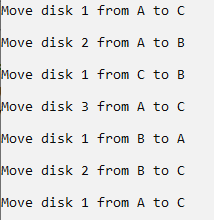
}

int main(){

toh(3 , 'A' , 'B' , 'C');

}

**Output :**

****

**PRACTICAL NO – 04**

**Implement the following for Queue**

**A) Write a program to implement the concept of Queue with Insert, Delete, Display and Exit operations**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

struct node{

int data;

struct node \*next;

};

struct node \*rear = NULL , \*front = NULL , \*temp = NULL;

int boundedCapacity = 5 , counter = 0 , val , chc;

void menu();

void continueUser(){

cout<<"\n1 ] Menu\t\t\t2 ] Exit\n------------------------------------------------------------\nChoice >>> ";

cin>>chc;

if(chc == 1){

menu();

}else if(chc == 2){

exit(0);

}

}

void enque(){

system("cls");

if(counter + 1 > boundedCapacity){

cout<<"\nQueue is in Overflow State !\n";

}else{

struct node \*element = (struct node \*)malloc(sizeof(struct node));

cout<<"\nNew Value : ";

cin>>val;

element -> data = val;

element -> next = NULL;

if(front == NULL && rear == NULL){

front = rear = element;

}else{

rear -> next = element;

rear = element;

}

counter++;

}

}

void display(){

if(counter == 0){

cout<<"\nQueue is in Underflow State !\n";

continueUser();

}

cout<<"\nAvailable Queue\n";

temp = front;

while(temp != rear){

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

temp = temp -> next;

}

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

continueUser();

}

void createQueue(){

system("cls");

cout<<"\nSize of Queue : ";

cin>>boundedCapacity;

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

enque();

}

cout<<"\nQueue created Successfully !\n";

continueUser();

}

void deque(){

if(counter == 0){

cout<<"\nQueue is in Underflow State !\n";

continueUser();

}else{

temp = front;

front = front -> next;

free(temp);

counter--;

cout<<"\nValue Deleted Successfully !\n";

}

continueUser();

}

void menu(){

system("cls");

cout<<"\n Queue Operations \n------------------------------------------------------------\n1 ] Create Queue\t\t\t2 ] Display\n\n3 ] Enque\t\t\t\t4 ] Deque\n\t\t\t5 ] Exit\n------------------------------------------------------------\nChoice >>> ";

cin>>chc;

system("cls");

switch(chc){

case 1 :

createQueue();

break;

case 2 :

display();

break;

case 3 :

enque();

continueUser();

break;

case 4 :

deque();

break;

case 5 :

exit(0);

break;

}

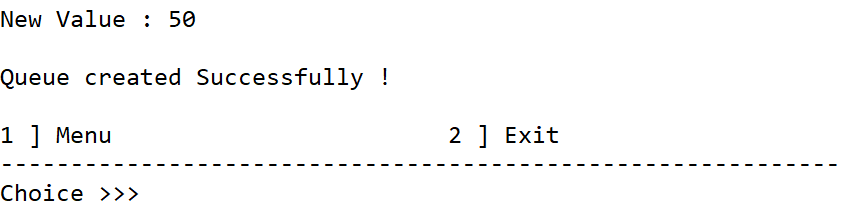
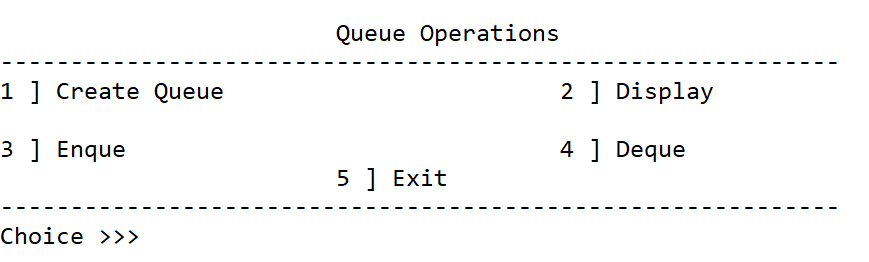
}

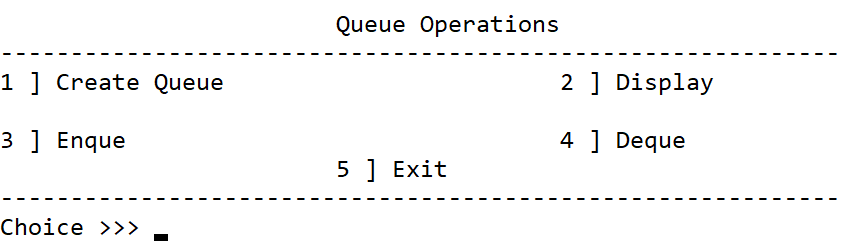
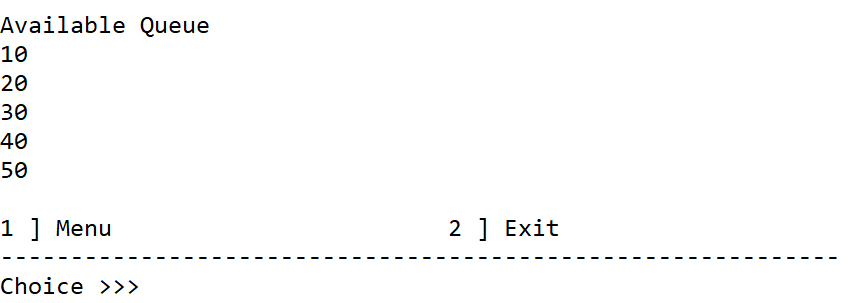
int main(){

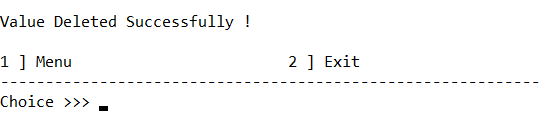
menu();

}

**Output :**

****

****

****

**B) Write a program to implement the concept of Circular Queue**

**Code :**

#include <iostream>

using namespace std;

int queue[10] , counter = 0 , boundedCapacity = 0 , i , front = -1 , rear = -1 , chc , value;

void menu();

userContinue(){

cout<<"\n1 ] Menu 2 ] Exit\n------------------------------\nChoice >>> ";

cin>>chc;

if(chc == 1){

menu();

}else if(chc == 2){

exit(0);

}

}

void createQueue(){

queue[boundedCapacity];

userContinue();

}

void displayQueue(){

if(counter == 0){

cout<<"\nQueue is in UnderFlow State\n";

}else{

cout<<"\nAvailable Queue\n";

if(front > rear){

for(i = front ; i < boundedCapacity ; i++){

cout<<queue[i]<<endl;

}

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

cout<<queue[i]<<endl;

}

}else{

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

cout<<queue[i]<<endl;

}

}

}

userContinue();

}

void EnQueue(){

cout<<"\nNew Value : ";

cin>>value;

if(counter == boundedCapacity){

cout<<"\nQueue is in OverFlow State\n";

}else{

if(counter == 0){

front = rear = 0;

}else{

rear = (rear + 1) % boundedCapacity;

}

queue[rear] = value;

counter++;

cout<<"\nValue Inserted Successfully\n";

}

userContinue();

}

void DeQueue(){

if(counter == 0){

cout<<"\nQueue is in UnderFlow State\n";

}else if(counter == 1){

counter--;

front = rear = -1;

cout<<"\nValue Deleted Successfully\n";

}else{

counter--;

front++;

cout<<"\nValue Deleted Successfully\n";

}

userContinue();

}

void menu(){

system("cls");

cout<<"\tCircular Queue\n------------------------------\n1 ] Create 2 ] Display\n\n3 ] EnQueue 4 ] DeQueue\n\n 5 ] Exit\n------------------------------\nChoice >>> ";

cin>>chc;

system("cls");

switch(chc){

case 1 :

cout<<"\nQueue Capacity : ";

cin>>boundedCapacity;

createQueue();

break;

case 2 :

displayQueue();

break;

case 3 :

EnQueue();

break;

case 4 :

DeQueue();

break;

case 5 :

exit(0);

break;

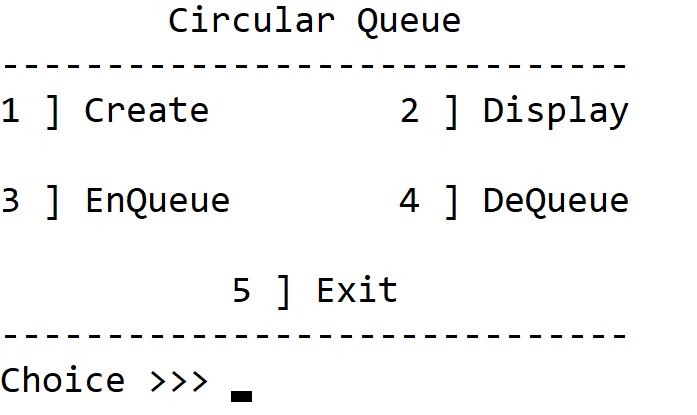
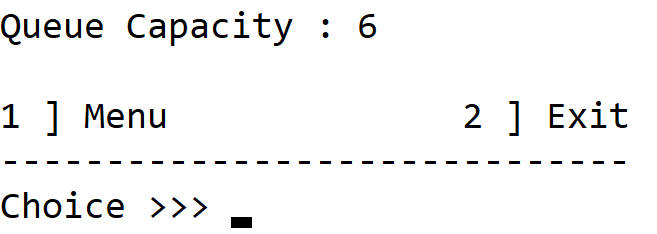
}

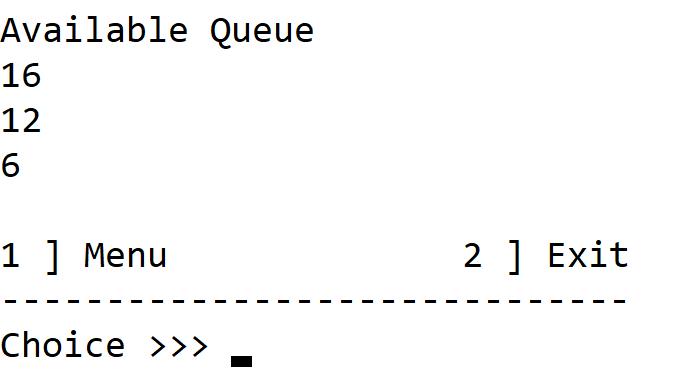
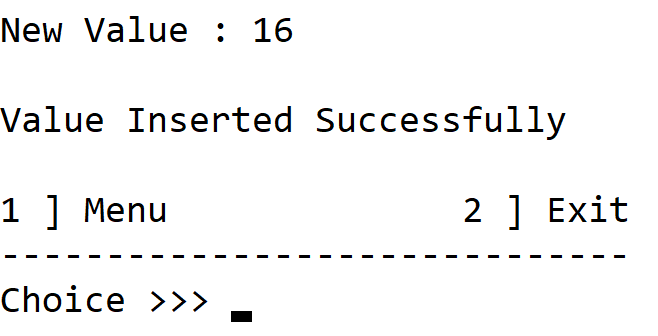
}

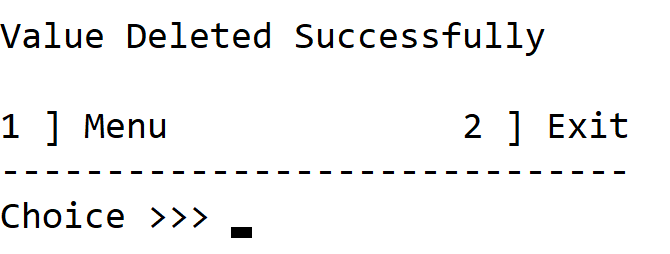
int main(){

menu();

}

**Output :**

****

****

**C) Write a program to implement the concept of Deque**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

struct node{

int data;

struct node \*next;

};

struct node \*rear = NULL , \*front = NULL , \*temp = NULL;

int boundedCapacity = 5 , counter = 0 , val , chc , i;

void menu();

void continueUser(){

cout<<"\n1 ] Menu\t\t\t2 ] Exit\n------------------------------------------------------------\nChoice >>> ";

cin>>chc;

if(chc == 1){

menu();

}else if(chc == 2){

exit(0);

}

}

void enque(){

system("cls");

if(counter + 1 > boundedCapacity){

cout<<"\nQueue is in Overflow State !\n";

}else{

struct node \*element = (struct node \*)malloc(sizeof(struct node));

cout<<"\nNew Value : ";

cin>>val;

element -> data = val;

element -> next = NULL;

cout<<"\n1 ] Enqueue at Rear \n2 ] Enqueue at Front \n\nChoice >>> ";

cin>>chc;

if(front == NULL && rear == NULL){

front = rear = element;

}else{

if(chc == 1){

element -> next = front;

front = element;

}else if(chc == 2){

rear -> next = element;

rear = element;

}

}

counter++;

}

}

void display(){

if(counter == 0){

cout<<"\nQueue is in Underflow State !\n";

continueUser();

}

cout<<"\nAvailable Queue\n";

temp = front;

while(temp != rear){

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

temp = temp -> next;

}

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

continueUser();

}

void createQueue(){

system("cls");

cout<<"\nBounded Capacity : ";

cin>>boundedCapacity;

menu();

}

void deque(){

if(counter == 0){

cout<<"\nQueue is in Underflow State !\n";

continueUser();

}else{

cout<<"\n1 ] Dequeue from Rear\n2 ] Dequeue from Front\n\nChoice >>> ";

cin>>chc;

if(chc == 1){

temp = front;

for(i = 1 ; i < counter - 2 ; i++){

temp = temp -> next;

}

free(temp -> next -> next);

temp -> next = NULL;

}else if(chc == 2){

temp = front;

front = front -> next;

free(temp);

}

counter--;

cout<<"\nValue Deleted Successfully !\n";

}

continueUser();

}

void menu(){

system("cls");

cout<<"\n Double Ended Queue\n------------------------------\n1 ] Create 2 ] Display\n\n3 ] EnQueue 4 ] DeQueue\n\n 5 ] Exit\n------------------------------\nChoice >>> ";

cin>>chc;

system("cls");

switch(chc){

case 1 :

createQueue();

break;

case 2 :

display();

break;

case 3 :

enque();

continueUser();

break;

case 4 :

deque();

break;

case 5 :

exit(0);

break;

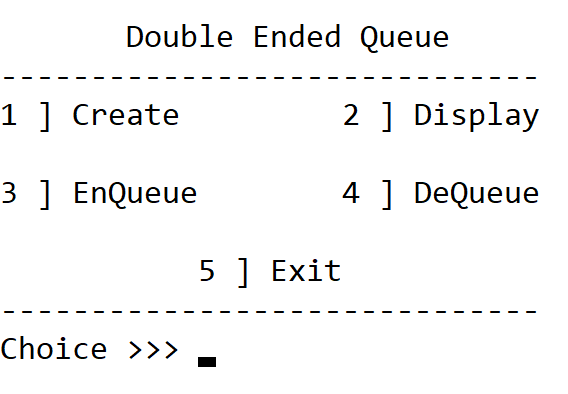
}

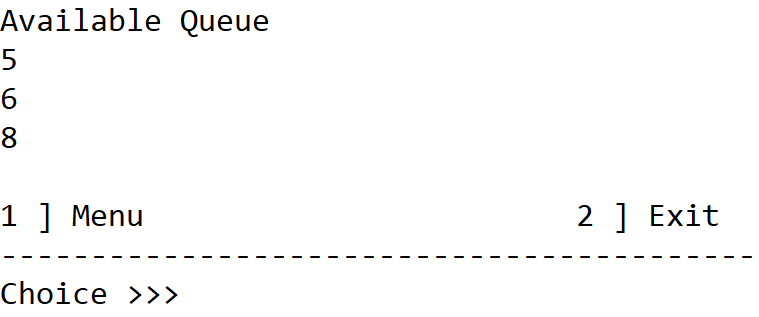
}

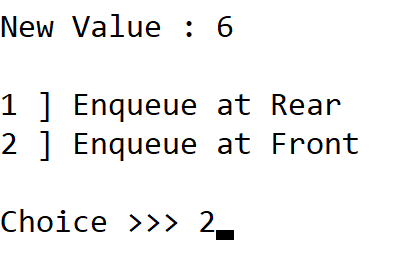
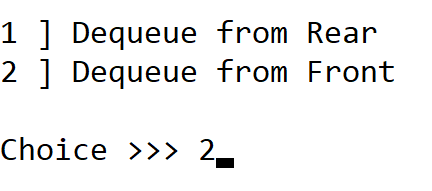
int main(){

menu();

}

**Output :**

****

****

**PRACTICAL NO – 05**

**Implement the following sorting techniques**

**A) Write a program to implement bubble sort**

**Code :**

#include <iostream>

using namespace std;

int main(){

int i , j , size , loop;

do{

system("cls");

cout<<"\nSize of Array : ";

cin>>size;

int ary[size];

for(i = 0 ; i < size ; i++){ //Array insertion

cout<<"Element "<<i + 1<<" : ";

cin>>ary[i];

}

system("cls");

cout<<"Given Array : ";

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

cout<<ary[i]<<" ";

}

for(i = 0 ; i < size - 1 ; i++){ // Sorting logic

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

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

int t = ary[j];

ary[j] = ary[j + 1];

ary[j + 1] = t;

}

}

}

cout<<endl<<"Sorted Array : ";

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

cout<<ary[i]<<" ";

}

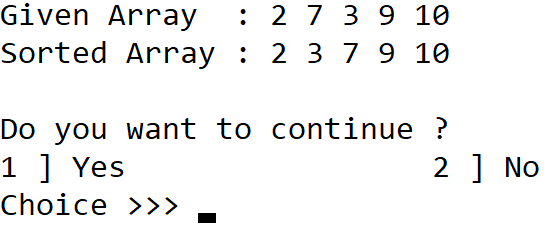
cout<<"\n\nDo you want to continue ? \n1 ] Yes\t\t\t2 ] No\nChoice >>> ";

cin>>loop;

}while(loop != 2);

}

**Output :**

****

**B) Write a program to implement selection sort**

**Code :**

#include <iostream>

using namespace std;

int main(){

int size , i , j , chc;

do{

system("cls");

cout<<"\nSize of Array : ";

cin>>size;

int ary[size];

for(i = 0 ; i < size ; i++){ // Array insertion

cout<<"Element "<<i + 1<<" : ";

cin>>ary[i];

}

system("cls");

cout<<"\nGiven Array : ";

for(i = 0 ; i < size ; i++){ // Array display

cout<<ary[i]<<" ";

}

cout<<"\nSorted Array : ";

for(i = 0 ; i < size - 1 ; i++){ // Sorting logic

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

if(ary[j] < ary[i]){

int t = ary[j];

ary[j] = ary[i];

ary[i] = t;

}

}

}

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

cout<<ary[i]<<" ";

}

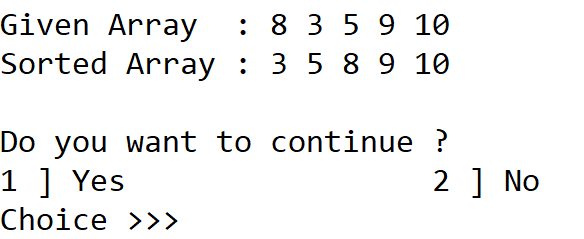
cout<<"\n\nDo you want to continue ?\n1 ] Yes\t\t\t2 ] No\nChoice >>> ";

cin>>chc;

}while(chc != 2);

}

**Output :**

****

**C) Write a program to implement insertion sort**

**Code :**

#include <iostream>

using namespace std;

int main(){

int size , i , j , chc , key;

do{

system("cls");

cout<<"\nSize of Array : ";

cin>>size;

int ary[size];

for(i = 0 ; i < size ; i++){ // Array insertion

cout<<"Element "<<i + 1<<" : ";

cin>>ary[i];

}

system("cls");

cout<<"\nGiven Array : ";

for(i = 0 ; i < size ; i++){ // Array display

cout<<ary[i]<<" ";

}

cout<<"\nSorted Array : ";

for(i = 1 ; i < size ; i++){ // Sorting logic

key = ary[i];

for(j = i - 1 ; (j >= 0 && ary[j] > key) ; j--){

ary[j + 1] = ary[j];

}

ary[j + 1] = key;

}

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

cout<<ary[i]<<" ";

}

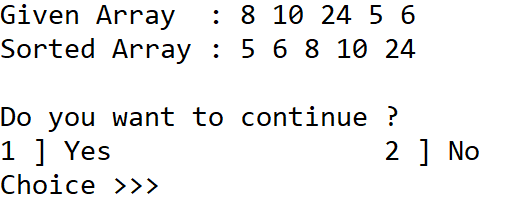
cout<<"\n\nDo you want to continue ?\n1 ] Yes\t\t\t2 ] No\nChoice >>> ";

cin>>chc;

}while(chc != 2);

}

**Output :**

****

**PRACTICAL NO – 06**

**Implement the following data structure techniques**

**A) Write a program to search the element using sequential search**

**Code :**

#include <iostream>

using namespace std;

int searchElement(int ary[] , int size , int item){

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

if(item == ary[i]){ // Checking if item is in array or not

return i;

}

}

return -1;

}

int main(){

int size , res , item , chc;

do{

system("cls");

cout<<"\nArray Size : ";

cin>>size;

int ary[size];

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

cout<<"Element "<<i + 1<<" : ";

cin>>ary[i];

}

system("cls");

cout<<"\nItem to search : "; // Taking user input

cin>>item;

system("cls");

cout<<"\nArray : ";

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

cout<<ary[i]<<" ";

}

res = searchElement(ary , size , item); // Storing the index number of item searched

if(res == -1){

cout<<"\nMatch not found !";

}else{

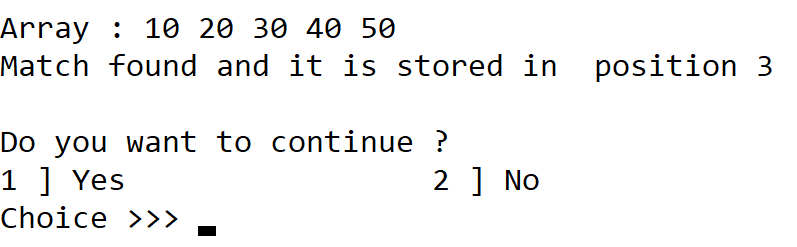
cout<<"\nMatch found and it is stored in "<<" position !"<<res + 1;

}

cout<<"\n\nDo you want to continue ?\n1 ] Yes\t\t\t2 ] No\nChoice >>> ";

cin>>chc;

}while(chc != 2);

****}

**Output :**

**B) Write a program to search the element using binary search**

**Code :**

#include <iostream>

#include <algorithm>

using namespace std;

int binarySearch(int ary[] , int arraySize , int item){

int startIndex = 0 , lastIndex = arraySize;

while(startIndex <= lastIndex){ // Loop runs until there is only 1 element remaining in these limits

int midPoint = (startIndex + lastIndex) / 2; // Calculating midpoint index every time the array decreases

if(ary[midPoint] == item){

return midPoint; // Returning the only one element remaining in the array limits

}else if(item < ary[midPoint]){

lastIndex = midPoint - 1; // Decreasing array if element is on left side of midpoint

}else if(ary[midPoint] < item){

startIndex = midPoint + 1; // Decreasing array if element is on right side of midpoint

}

}

return -1;

}

int main(){

int chc , item , arraySize , res;

do{

system("cls");

cout<<"\nArray Size : ";

cin>>arraySize;

int ary[arraySize];

for(int i = 0 ; i < arraySize ; i++){ // Array insertion

cout<<"Element "<<i + 1<<" : ";

cin>>ary[i];

}

sort(ary , ary + arraySize); // Sorting array

system("cls");

cout<<"\nItem to find : ";

cin>>item;

cout<<"\nArray : ";

for(int i = 0 ; i < arraySize ; i++){ // Array printing

cout<<ary[i]<<" ";

}

res = binarySearch(ary , arraySize , item);

if(res == -1){

cout<<"\nMatch not found !";

}else{

cout<<"\nMatch found and it is stored in position "<<res + 1;

}

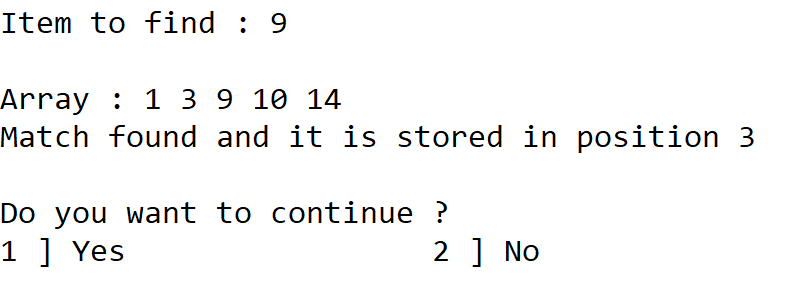
cout<<"\n\nDo you want to continue ?\n1 ] Yes\t\t\t2 ] No";

cin>>chc;

}while(chc != 2);

}

**Output :**

****

**PRACTICAL NO – 07**

**Implement the following data structure techniques**

**A) Write a program to create the tree and display the elements**

**Code :**

#include <iostream>

#include <stdlib.h>

using namespace std;

int chc , val , size , i;

struct node{

int value;

struct node \*leftChild , \*rightChild;

};

void menu();

struct node \*createNode(int value){

if(value == -1){

return 0;

}

struct node \*newNode = (struct node \*)malloc(sizeof(struct node));

newNode -> leftChild = NULL;

newNode -> rightChild = NULL;

newNode -> value = value;

return newNode;

}

void inorderTraversal(struct node \*root){

if(root == NULL){

return;

}

inorderTraversal(root -> leftChild);

cout<<root -> value<<" ";

inorderTraversal(root -> rightChild);

}

void menu(){

do{

system("cls");

cout<<"\t\tTree Data Structure\n1 ] Create Tree\n2 ] Display(inorder)\n3 ] Exit\n\nChoice >>> ";

cin>>chc;

system("cls");

struct node \*root;

switch(chc){

case 1 :

cout<<"Data (-1 for no node) : ";

cin>>val;

root = createNode(val);

cout<<"Data for "<<root -> value<<"'s left child (-1 for no node) : ";

cin>>val;

root -> leftChild = createNode(val);

cout<<"Data for "<<root -> value<<"'s right child (-1 for no node) : ";

cin>>val;

root -> rightChild = createNode(val);

break;

case 2 :

inorderTraversal(root);

break;

case 3 :

exit(0);

}

cout<<"\nDo you want to continue ?\n[ 1 ] Yes\t\t[ 2 ] No\nChoice >>> ";

cin>>chc;

}while(chc != 2);

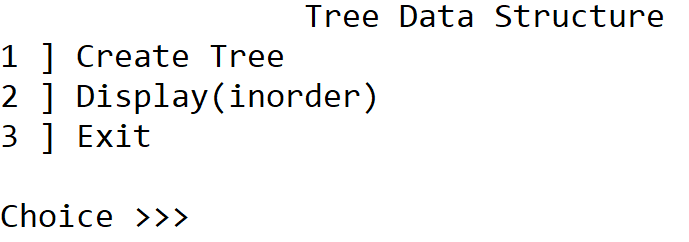
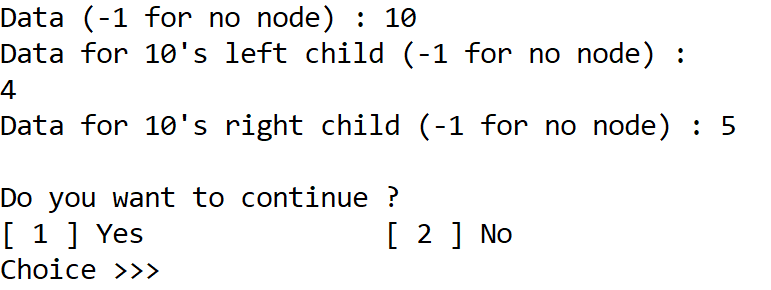
}

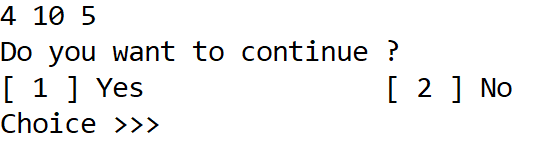
int main(){

menu();

}

**Output :**

**** ****

****

**B) Write a program to construct the binary tree**

**Code :**

**Output :**

**C) Write a program for inorder, postorder and preorder traversal of tree**

**Code :**

**Output :**