

**Data Structure Lab Using C/C++ MCA Sem 1**

**Name : Vaishnavi vidyadhar kothawade Division :** B

**Batch :** B2 **Semester :** MCA sem-1 **Roll No : 32**

Page no. 1



Bharati Vidyapeeth’s

**Institute of Management & Information Technology**

C.B.D. Belapur, Navi Mumbai 400614

Vision:

## Providing high quality, innovative and value-based education in information technology to build competent professionals.

Mission

M1. Technical Skills:-To provide solid technical foundation theoretically as well as practically capable of providing quality services to industry.

M2. Development: -Department caters to the needs of students through comprehensive educational programs and promotes lifelong learning in the field of computer Applications.

M3. Ethical leadership:-Department develops ethical leadership insight in the students to succeed in industry, government and academia.

CERTIFICATE

**This is to certify that the journal is the work of Miss. Vaishnavi Vidyadhar Kothawade, Roll No. 32 of MCA (**Sem- 1 Div: B **) for the academic year 2023 – 2024**

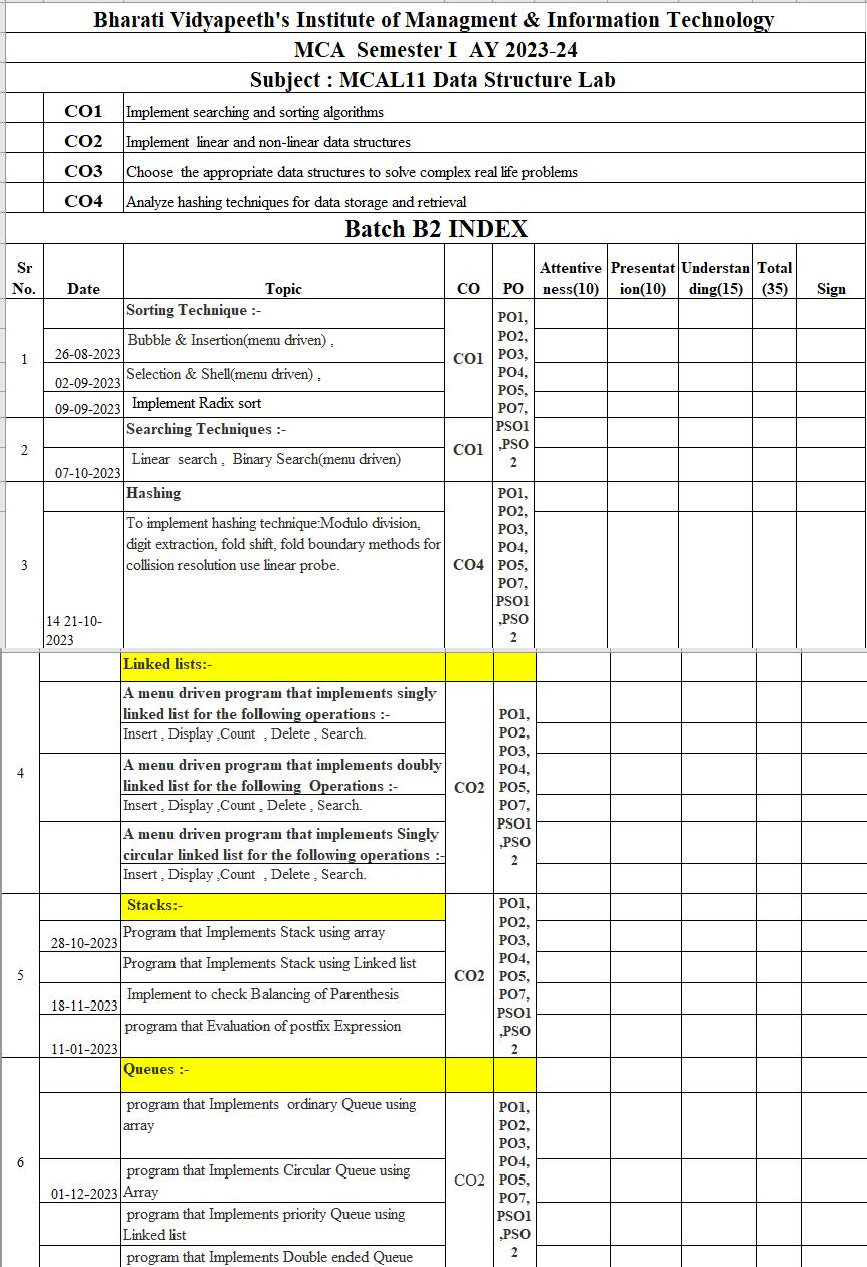
**Subject Code: MCAE251**

# Subject Name: Data Structure Lab Using C/C++

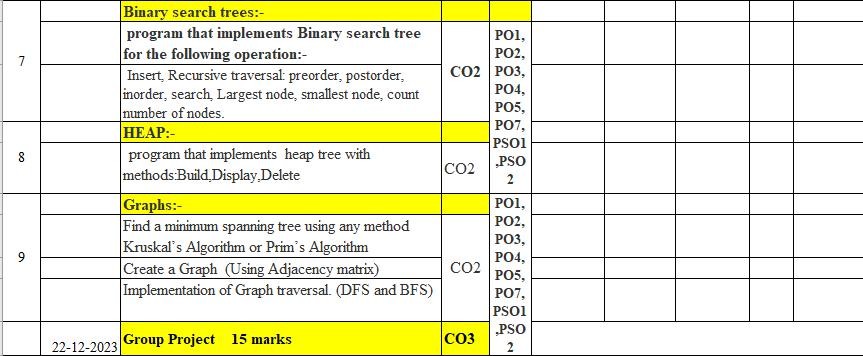
**Subject-in-charge Principal**

**Date:**

Page no. 2



Page no. 3



Page no. 4

## Sorting Techniques:

### Bubble and Insertion Sort

#include<iostream.h> #include<conio.h>

void swap(int &a, int &b) { int temp = a;

a = b;

b = temp;

}

void selectionSort(int arr[], int n) { for (int i = 0; i < n - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < n; j++) { if (arr[j] < arr[minIndex]) {

minIndex = j;

}}

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

}}

void bubbleSort(int arr[], int n) { for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) { if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}}}}

void displayArray(int arr[], int n) { cout << "Sorted Array: ";

for (int i = 0; i < n; i++) { cout << arr[i] << " ";

}

cout << endl;

}

int main() { clrscr();

while (1) { int n;

Page no. 5

cout << "Enter the number of elements in the array: "; cin >> n;

int arr[100];

cout << "Enter the elements of the array: "; for (int i = 0; i < n; i++) {

cin >> arr[i];

}

int choice;

cout << "\nChoose Sorting Algorithm:\n"; cout << "1. Selection Sort\n";

cout << "2. Bubble Sort\n";

cout << "Enter your choice (1 or 2): "; cin >> choice;

switch (choice) { case 1:

selectionSort(arr, n); break;

case 2:

bubbleSort(arr, n); break;

default:

cout << "Invalid choice!"; return 1;

}

displayArray(arr, n); char cont;

cout << "\nDo you want to continue? (y/n): "; cin >> cont;

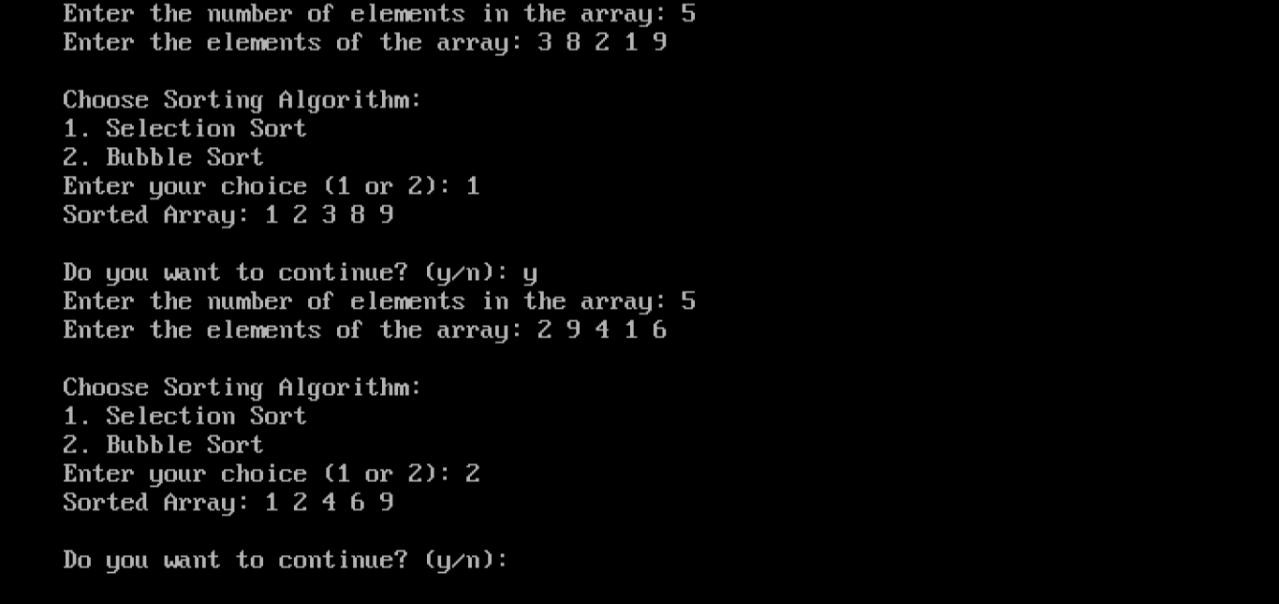
if (cont != 'y' && cont != 'Y') { break;

}}

getch(); return 0;

}

Page no. 6



### Selection and Shell Sort

#include<iostream.h> #include<conio.h>

void swap(int &a, int &b) { int temp = a;

a = b;

b = temp;

}

void selectionSort(int arr[], int n) { for (int i = 0; i < n - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < n; j++) { if (arr[j] < arr[minIndex]) {

minIndex = j;

}}

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

}}

void shellSort(int arr[], int n) {

for (int gap = n / 2; gap > 0; gap /= 2) { for (int i = gap; i < n; i++) {

int temp = arr[i], j;

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) { arr[j] = arr[j - gap];

}

arr[j] = temp;

Page no. 7

}}}

void displayArray(int arr[], int n) { cout << "Sorted Array: ";

for (int i = 0; i < n; i++) { cout << arr[i] << " ";

}

cout << endl;

}

int main() { clrscr();

while (1) { int n;

cout << "Enter the number of elements in the array: "; cin >> n;

int arr[100];

cout << "Enter the elements of the array: "; for (int i = 0; i < n; i++) {

cin >> arr[i];

}

int choice;

cout << "\nChoose Sorting Algorithm:\n"; cout << "1. Selection Sort\n";

cout << "2. Shell Sort\n";

cout << "Enter your choice (1 or 2): "; cin >> choice;

switch (choice) { case 1:

selectionSort(arr, n); break;

case 2:

shellSort(arr, n); break;

default:

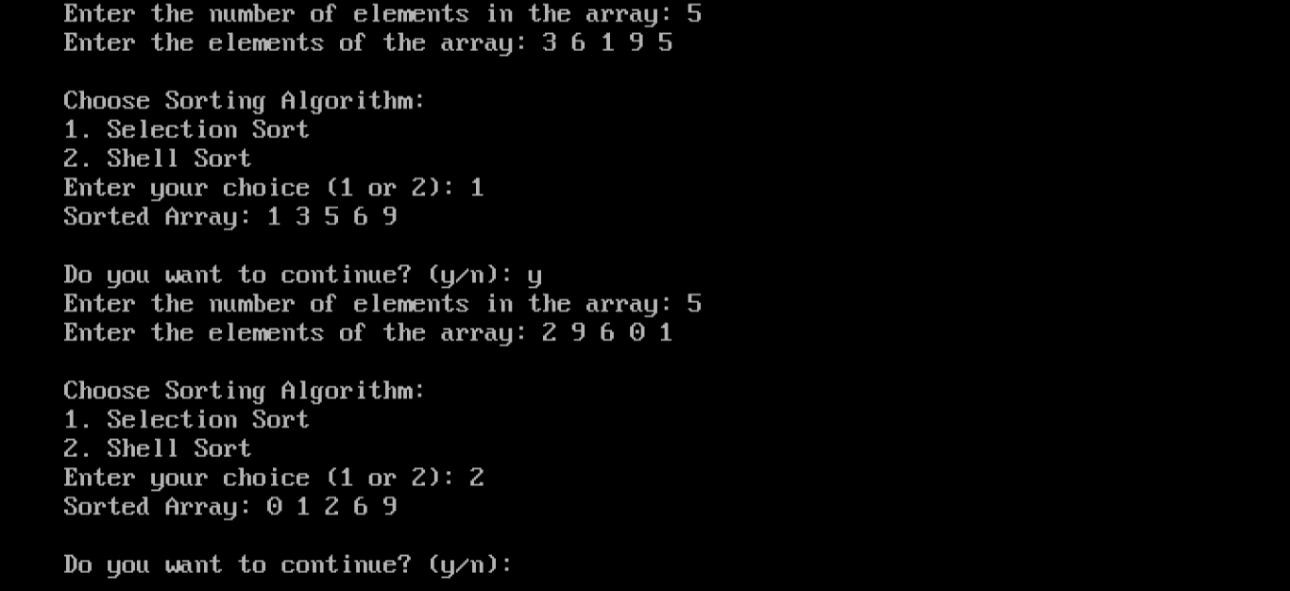
cout << "Invalid choice!"; return 1;

}

displayArray(arr, n);

Page no. 8

char cont;



cout << "\nDo you want to continue? (y/n): "; cin >> cont;

if (cont != 'y' && cont != 'Y') { break;

}}

getch(); return 0;

}

### Radix Sort

#include<iostream.h> #include<conio.h> class radix

{

int A[20],n,i; public:

void getdata();

void radsort(int\*,int); void display(int\*,int);

};

void radix::getdata()

{

cout<<"\n\t\tRadix Sorting"; cout<<"\n\t Enter the size of array:"; cin>>n;

Page no. 9

cout<<"\n\t Enter the elements to sort:\n"; for(i=0;i<n;i++)

{

cout<<"\t:"; cin>>A[i];

}

radsort(A,n);

display(A,n);

}

void radix::radsort(int \*A,int n)

{

int temp;

int bucket[10][20]; int buck\_count[10];

int i,k,j,r,no\_of\_passes=0,divisor=1; int largest,pass\_no;

largest=A[0];

for(i=1;i<n;i++)//find largest element

{

if(A[i]>largest) largest=A[i];

}

while(largest>0)//find passes find no of digits in number

{

no\_of\_passes++; largest=largest/10;

}

for(pass\_no=0;pass\_no<no\_of\_passes;pass\_no++)

{

for(k=0;k<10;k++)//initialize the buckets buck\_count[k]=0;//bucket count for(i=0;i<n;i++)

{

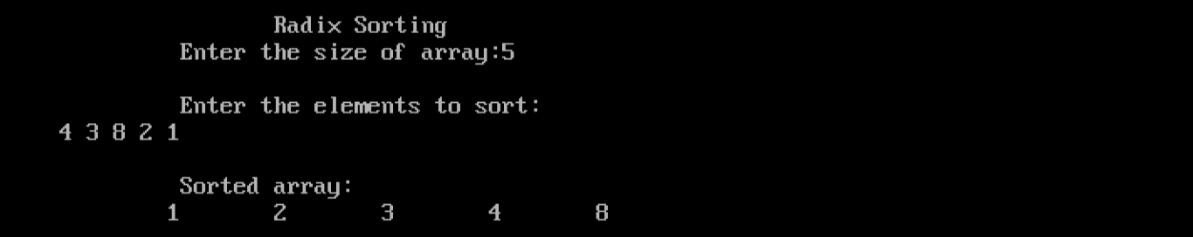
r=(A[i]/divisor)%10; bucket[r][buck\_count[r]++]=A[i];

}

i=0;//collect elements from buckets

Page no. 10

for(k=0;k<10;k++)



{

for(j=0;j<buck\_count[k];j++) A[i++]=bucket[k][j];

}

divisor=divisor\*10;

}}

void radix::display(int\* A,int n)

{

int i;

cout<<"\n\t Sorted array:\n\t"; for(i=0;i<n;i++) cout<<A[i]<<"\n\t";

}

void main()

{

radix r; clrscr();

r.getdata();

getch();

}

Page no. 11

## Searching Techniques:

### Linear and Binary Search

#include<iostream.h> #include<conio.h>

int linearSearch(int arr[], int n, int key) { for (int i = 0; i < n; i++) {

if (arr[i] == key) {

return i; // Return the index where the key is found

}}

return -1; // Return -1 if the key is not found

}

int binarySearch(int arr[], int low, int high, int key) { while (low <= high) {

int mid = (low + high) / 2; if (arr[mid] == key) {

return mid; // Return the index where the key is found

} else if (arr[mid] < key) { low = mid + 1;

} else {

high = mid - 1;

}}

return -1; // Return -1 if the key is not found

}

void displayResult(int index) { if (index != -1) {

cout << "Element found at index: " << index << endl;

} else {

cout << "Element not found in the array." << endl;

}}

int main() { clrscr(); int n;

cout << "Enter the number of elements in the array: "; cin >> n;

int arr[100];

cout << "Enter the elements of the sorted array: ";

Page no. 12

for (int i = 0; i < n; i++) { cin >> arr[i];

}

int choice, key; while (1) {

cout << "\nChoose Search Algorithm:\n"; cout << "1. Linear Search\n";

cout << "2. Binary Search\n";

cout << "Enter your choice (1 or 2): "; cin >> choice;

switch (choice) { case 1:

cout << "Enter the key to search: "; cin >> key;

displayResult(linearSearch(arr, n, key)); break;

case 2:

cout << "Enter the key to search: "; cin >> key;

displayResult(binarySearch(arr, 0, n - 1, key)); break;

default:

cout << "Invalid choice!"; return 1;

}

char cont;

cout << "\nDo you want to continue? (y/n): "; cin >> cont;

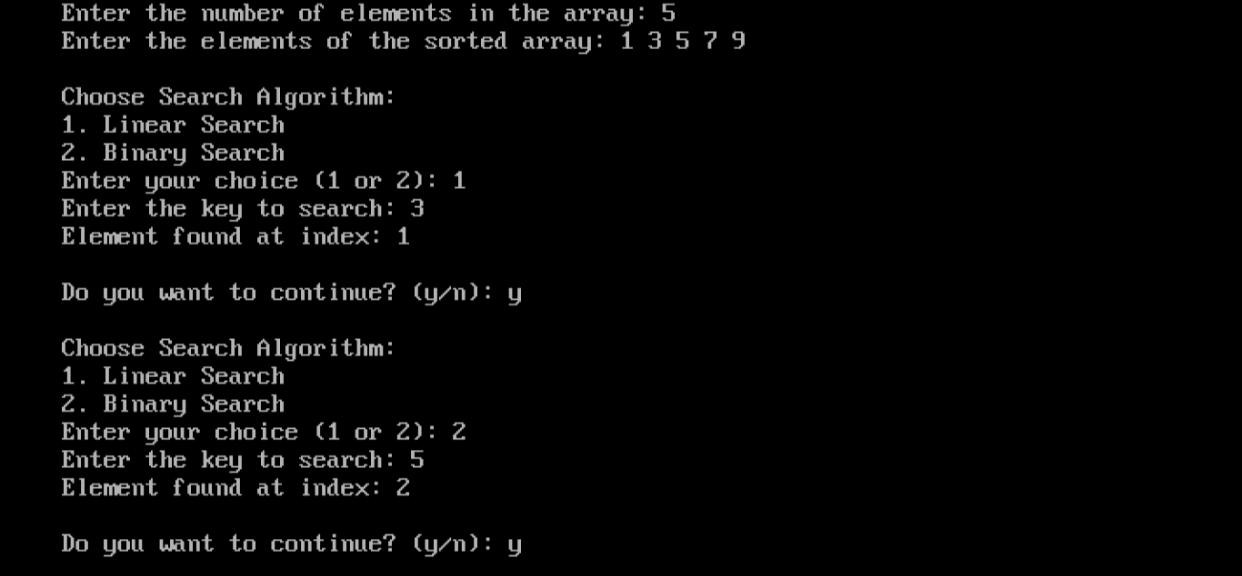
if (cont != 'y' && cont != 'Y') { break;

}}

getch(); return 0;

}

Page no. 13



Page no. 14

1. Hashing: #include<iostream.h> #include<conio.h>

const int TABLE\_SIZE = 10; class HashTable {

private:

int table[TABLE\_SIZE]; int linearProbe;

// Hashing methods

int moduloDivision(int key) { return key % TABLE\_SIZE;

}

int digitExtraction(int key) {

return key % 10; // Extract the last digit as the hash value

}

int foldShift(int key) { int sum = 0;

while (key > 0) { sum += key % 10; key /= 10;

}

return sum % TABLE\_SIZE;

}

int foldBoundary(int key) { int sum = 0;

int numDigits = 0;

while (key > 0) { sum += key % 10; key /= 10; numDigits++;

Page no. 15

}

// Ensure that the hash value is within the table size return (sum + numDigits) % TABLE\_SIZE;

}

int linearProbeHash(int hashValue) {

return (hashValue + linearProbe) % TABLE\_SIZE;

}

public:

HashTable() {

for (int i = 0; i < TABLE\_SIZE; i++) {

table[i] = -1; // Initialize table with -1 to indicate empty slots

}

}

void insert(int key) { int hashValue;

cout << "\nEnter linear probe value: "; cin >> linearProbe;

// Choose hashing method int choice;

cout << "\nChoose Hashing Method:\n"; cout << "1. Modulo Division\n";

cout << "2. Digit Extraction\n"; cout << "3. Fold Shift\n";

cout << "4. Fold Boundary\n";

cout << "Enter your choice (1-4): "; cin >> choice;

switch (choice) { case 1:

hashValue = moduloDivision(key); break;

Page no. 16

case 2:

hashValue = digitExtraction(key); break;

case 3:

hashValue = foldShift(key); break;

case 4:

hashValue = foldBoundary(key); break;

default:

cout << "Invalid choice!"; return;

}

// Linear probing if the slot is occupied while (table[hashValue] != -1) {

hashValue = linearProbeHash(hashValue);

}

// Insert the key into the table table[hashValue] = key;

cout << "Key " << key << " inserted at index " << hashValue << endl;

}

void display() {

cout << "\nHash Table:\n";

for (int i = 0; i < TABLE\_SIZE; i++) { if (table[i] != -1) {

cout << "Index " << i << ": " << table[i] << endl;

} else {

cout << "Index " << i << ": Empty" << endl;

}

}

}

};

Page no. 17

int main() { clrscr();

HashTable hashTable; char cont;

do {

cout << "\n1. Insert Key\n";

cout << "2. Display Hash Table\n"; cout << "3. Exit\n";

cout << "Enter your choice (1-3): "; int choice;

cin >> choice;

switch (choice) { case 1:

int key;

cout << "Enter the key to insert: "; cin >> key; hashTable.insert(key);

break; case 2:

hashTable.display(); break;

case 3:

cout << "Exiting program.\n"; getch();

return 0; default:

cout << "Invalid choice! Please enter a valid option.\n";

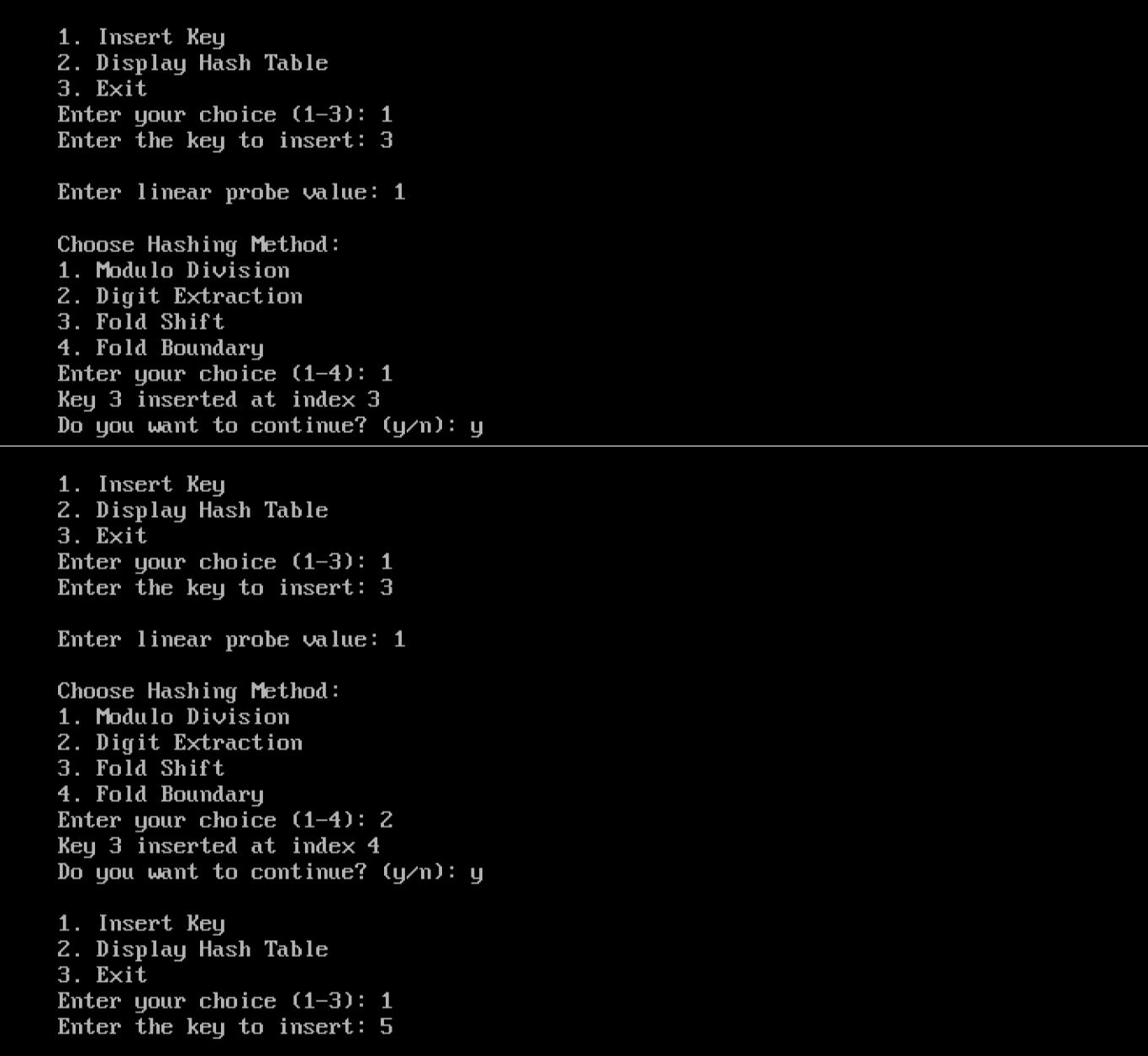
}

cout << "Do you want to continue? (y/n): "; cin >> cont;

} while (cont == 'y' || cont == 'Y');

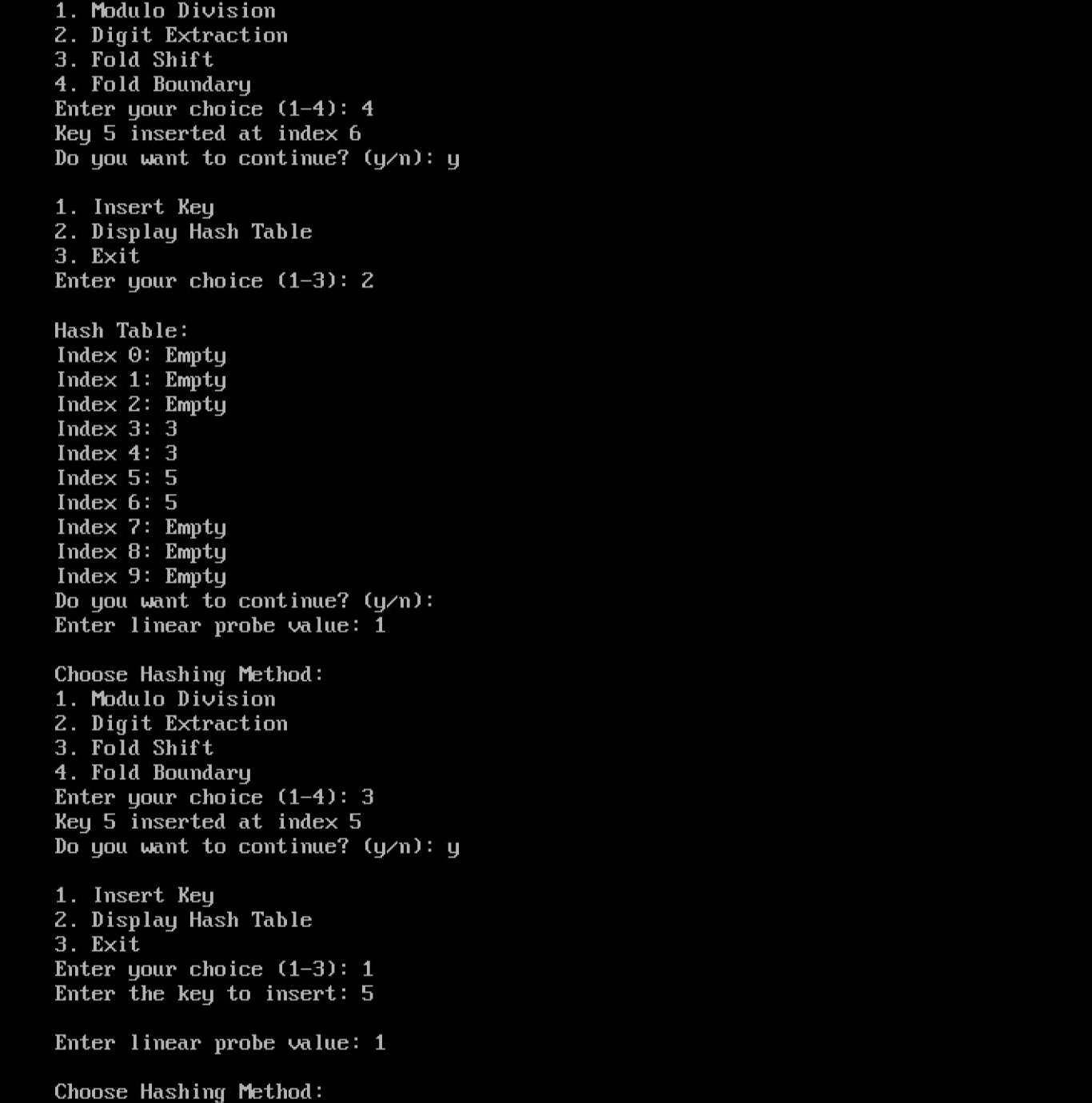
Page no. 18

getch(); return 0;



}

Page no. 19



Page no. 20

## Linked List:

**Singly Linked List** #include<iostream.h> #include<conio.h>

class node {

private:

public:

};

node \* p; node \* q;

void node :: add(int num)

{

int data;

node \*address;

int count(void); void add(int);

void display(void); void sort(void); void insert(int,int); void remove(int); void search(int);

q=p; if(p==NULL)

{

}

else

{

p=new node; p-> data=num;

p-> address=NULL;

while(q->address!=NULL)

{

q=q->address;

}

q->address=new node; q->address->data=num;

q->address->address=NULL;

}

}

void node :: display(void)

{

q=p;

if( p == NULL)

{

}else

{

cout<< "No Linkedlist";

while (q != NULL)

Page no. 21

{

cout<<q->data<<endl; q = q->address;

}

}

}

int node :: count(void)

{

q=p;

int i = 0;

if(p == NULL)

{

}

else

{

return 0;

while (q!= NULL)

{

}

return i;

}

}

i++;

q = q-> address;

void node :: sort()

{

//node \* q = p; node \*i;

node \*j; int temp;

for(i=p;i!=NULL;i=i->address)

{

for(j=i->address;j!=NULL;j=j->address)

{

if(i->data>j->data)

{

temp=i->data;

i->data=j->data; j->data=temp;

}

}

}

}

void node :: insert(int pos, int num)

{

node \*q = p; node \*temp; int i;

if(pos == 1)

{

p = new node;

Page no. 22

p ->data=num; p->address = q; return;

}

if(pos == 1+count())

{

add(num); return;

}

for(i=1; i<=(pos-2);i++)

{

q=q->address;

}

temp=q->address;

q->address = new node; q->address->data = num;

q->address->address=temp;

}

void node :: remove(int pos)

{

node \*temp; node \*q=p; if(pos == 1)

{

p=q->address; delete(q); return;

}

for(int i=1; i<=(pos-2);i++)

{

q=q->address;

}

temp = q->address;

q->address = q->address->address; delete (temp);

}

void node :: search(int num)

{

node \*q=p; int flag=0; int pos=0;

for(q=p;q!=NULL;q=q->address)

{

if(q->data == num)

{

}

pos++;

}

flag=1; break;

if(flag==1)

Page no. 23

{

}

else

{

}}

void main()

{

cout<<"Number Found At Position : "<<pos;

cout<<"Number NA Found";

int num, option,pos; node n;

p=NULL;

char ch='y'; clrscr(); while(ch =='y')

{

cout<<"\n1.Add"; cout<<"\n2.Display"; cout<<"\n3.Count"; cout<<"\n4.Sort"; cout<<"\n5.Insert"; cout<<"\n6.Delete"; cout<<"\n7.Search"; cout<<"\nEnter an option: "; cin>>option;

switch(option)

{

case 1:

{

}

case 2:

{

}

case 3:

{

}

case 4:

{

cout<<"\nEnter the value: "; cin>>num;

n.add(num);

cout<<"\nDo you Want to do again\n"; break;

n.display();

cout<<"\nDo you Want to do again\n"; break;

num=n.count(); cout<<num;

cout<<"\nDo you Want to do again\n"; break;

cout<<"Sorted List";

Page no. 24

}

case 5:

{

}

case 6:

{

}

case 7:

{

}

}

ch=getch(); } }

### Doubly Linked List

# include<iostream.h> # include<conio.h> class node

{

n.sort();

cout<<"\nDo you want to continue\n"; n.display();

break;

cout<<"Enter Position: "; cin>>pos;

cout<<"Enter Number: "; cin>>num; n.insert(pos,num);

cout<<"\nDo you Want to do again\n"; break;

cout<<"Enter Position"; cin>>pos; n.remove(pos);

cout<<"\nDo you Want to do again\n"; break;

cout<<"Enter Number: "; cin>>num; n.search(num);

cout<<"\nDo you Want to do again\n"; break;

private:

int data; node \*next; node \*prev; public:

void add(int);

void display(void); int count(void); void sort(void); void insert(int,int); void remove(int);

Page no. 25

};

node \*p;

void search(int); void reverse(void);

void node::add(int num)

{

node \*q=p;

if(p==NULL)

{

}

else

{

}

}

p=new node; p->data=num;

p->next=NULL; p->prev=NULL;

while(q->next != NULL)

{

q=q->next;

}

q->next=new node; q->next->prev =q;

q->next->data=num; q->next->next=NULL;

void node::display(void)

{

node \*q=p; if(p==NULL)

{

}

else

{

cout<<"No Elements link list/n";

while(q!=NULL)

{

cout<<q->data; q=q->next;

}

}

}

int node::count(void)

{

node \*q=p; int i=0; if(p==NULL)

{

return 0;

}

Page no. 26

else

{

while(q!=NULL)

{

}

return i;

}

}

i=i+1;

q=q->next;

void node:: sort(void)

{

node \*i; node \*j; int temp;

for(i=p;i!=NULL;i=i->next)

{

for(j=i->next;j!=NULL;j=j->next)

{

if(i->data>j->data)

{

temp=i->data;

i->data=j->data; j->data=temp;

}

}

}

}

void node:: insert(int pos,int num)

{

node\*temp; node \*q=p; int i; if(pos==1)

{

p=new node; p->data=num; p->next=q;

p->prev=NULL; return;

}

if(pos==1+count())

{

add(num); return;

}

temp=q->next; for(i=1;i<=(pos-2);i++)

{

q=q->next;

}

Page no. 27

q->next=new node; q->next->data=num; q->next->prev=q;

q->next->next=temp;

q->next->next->prev=q->next;

}

void node::remove(int pos)

{

int i; node\*q=p;

node\*temp; if(pos==1)

{

p=q->next; delete (q);

p->prev=NULL; return;

}

for(i=1;i<=(pos-2);i++)

{

q=q->next;

}

temp=q->next;

q->next=q->next->next; q->next->prev=q; delete(temp);

}

void node:: search(int num)

{

node\*q= p; int flag =0;

int pos =0; for(q=p;q!=NULL;q=q->next)

{

if(q->data == num){ flag =1; break;

}

pos++;

}

if(flag ==1)

{

cout<<"Number found at position"<<pos;

}

else

{

}

}

cout<<"number not found";

Page no. 28

void node::reverse(void)

{

node \*q=p;

if(p==NULL)

{

cout<<"NO LinkList";

}

else

{

while(q->next!=NULL)

{

q=q->next;

}

while(q!=NULL)

{

cout<<q->data<<endl; q=q->prev;

}

}

}

void main(void)

{

int num,pos,option; node n;

p=NULL;

char ch='y'; clrscr(); while(ch=='y')

{

cout<<"\n 1.Add"; cout<<"\n 2.Display"; cout<<"\n 3.Count"; cout<<"\n 4.Sort"; cout<<"\n 5.Insert"; cout<<"\n 6.Remove"; cout<<"\n 7.Search"; cout<<"\n 8.Reverse "; cout<<"\n Enter Option : " ; cin>>option;

switch(option)

{

case 1:

{

}

cout<<"\nEnter the value: "; cin>>num;

n.add(num);

cout<<"\nDo you Want to do again\n"; break;

Page no. 29

case 2:

{

}

n.display();

cout<<"\nDo you Want to do again\n"; break;

case 3:

{

}

case 4:

{

}

case 5:

{

}

case 6:

{

}

case 7:

{

}

case 8:

{

num=n.count(); cout<<num;

cout<<"\nDo you Want to do again\n"; break;

cout<<"Sorted List"; n.sort();

n.display();

cout<<"\nDo you want to continue\n"; break;

cout<<"Enter Position: "; cin>>pos;

cout<<"Enter Number: "; cin>>num; n.insert(pos,num);

cout<<"\nDo you Want to do again\n"; break;

cout<<"\nEnter Position: "; cin>>pos;

n.remove(pos);

cout<<"\nDo you Want to do again\n"; break;

cout<<"Enter Number: "; cin>>num; n.search(num);

cout<<"\nDo you Want to do again\n"; break;

cout<<"\nReverse List: \n";

Page no. 30

n.reverse();

cout<<"\nDO YOU WANT TO CONTINUE \n";

break;

}

}

ch=getch();

}

}

### Circular Linked List

#include<iostream.h> #include<conio.h>S class node

{

};

node \*p;

private:

int data;

node \*address; public:

void add(int);

void display(void); int count(void); void sort(void); void insert(int,int); void remove(int); void search(int);

void node::add(int num)

{

node \*q=p;

if(p==NULL)

{

}

else

{

}

p=new node; p->data=num; p->address=p;

while(q->address != p)

{

q=q->address;

}

q->address=new node; q->address->data=num; q->address->address=p;

Page no. 31

}

void node::display(void)

{

node \*q=p; if(p==NULL)

{

}

else

{

cout<<"No Circular link list/n";

do{

}

cout<<q->data; q=q->address;

while(q!= p);

}

}

int node::count(void)

{

node \*q=p; int i=0; if(p==NULL)

{

}

else

{

do{

return 0;

i=i++;

q=q->address;

}while(q!=p); return i;

}

}

void node:: sort(void)

{

node \*i; node \*j; int temp;

if(p==NULL)

{

cout<<"\n No link list";

}

else

{

for(i=p;i!=NULL;i=i->address)

{

for(j=i->address; j!=NULL; j=j->address)

{

if(i->data>j->data)

Page no. 32

{

temp=i->data;

i->data=j->data; j->data=temp;

}

}

}

}

}

void node:: insert(int pos,int num)

{

node\*temp; node \*q=p; int i; if(pos==1)

{

p=new node; p->data=num; p->address=q; return;

}

if(pos==1+count())

{

add(num); return;

}

for(i=1;i<=(pos-2);i++)

{

q=q->address;

}

temp=q->address;

q->address=new node; q->address->data=num;

q->address->address=temp;

}

void node::remove(int pos)

{

int i; node\*q = p; node\*temp;

if(pos==1)

{

p=q->address; delete (q); return;

}

for(i=1;i<=(pos-2);i++)

{

q=q->address;

}

Page no. 33

temp=q->address;

q->address=q->address->address; delete(temp);

}

void node:: search(int num)

{

do{

node\*q= p; int flag =0;

int pos =0;

if(q->data == num){ flag =1; break;

}

pos++;

q=q->address;

}while(q!=p);

if(flag ==1)

{

}

else

{

}

}

cout<<"Number found at position"<<pos;

cout<<"number not found";

void main(void)

{

int num,pos,option; node n;

p=NULL;

char ch='y'; clrscr(); while(ch=='y')

{

cout<<"\n 1.Add"; cout<<"\n 2.Display"; cout<<"\n 3.Count"; cout<<"\n 4.Sort"; cout<<"\n 5.Insert"; cout<<"\n 6.Remove"; cout<<"\n 7.Search";

cin>>option; switch(option)

{

case 1:

{

cout<<"\nEnter Number"; cin>>num;

n.add(num);

Page no. 34

}

case 2:

{

}

case 3:

{

cout<<"\nDo you want to Continue"; break;

n.display();

cout<<"\nDo you want to Continue"; break;

num=n.count();

cout<<"\ncount of linklist is= "<<num; cout<<"\nDo you want to continue"; break;

}

case 4:

{

n.sort();

cout<<"\n is sorted";

cout<<"\n Do you want to continue..."; break;

}

case 5:

{

}

case 6:

{

}

cout<<"\nENter position "; cin>>pos;

cout<<"\nEnter the Number "; cin>>num; n.insert(pos,num);

cout<<"\n Do you want to continue "; break;

cout<<"\nEnterPosition "; cin>>pos; n.remove(pos);

cout<<"Do You Want to continue "; break;

case 7:

{

cout<<"\nEnter Number to be searched"; cin>>num;

n.search(num);

cout<<"\nDo you wantto continue"; break;

}

Page no. 35

default: cout<<"Not valid";

}

ch=getch();

}

}

## Stack:

### Stack using Array

#include<stdio.h> #include<conio.h> #include<iostream.h> class stack

{

int data; stack \*next; stack \*pre; public:

void push(int); void display(void); int pop(void);

};

stack \*top; stack \*bottom;

void stack::push(int num)

{

if((top==NULL)&& (bottom==NULL))

{

top=bottom=new stack; top->data=num;

top->next=NULL; top->pre=NULL;

}

else

{

top->next=new stack; top->next->data=num; top->next->pre=top; top->next->next=NULL; top=top->next;

}

}

void stack::display()

{

Page no. 36

{

}

else

{

stack \*q=bottom; if((top==NULL)&&(bottom==NULL))

cout<<"\nNo stack exists.";

while(q!=NULL)

{

cout<<q->data<<" "; q=q->next;

}

}

}

int stack::pop()

{

{

}

{

}

else

{

int num; if((top==NULL)&&(bottom==NULL))

return -1;

num=top->data; top=top->pre; if(top!=NULL)

delete(top->next); top->next=NULL;

delete(bottom); bottom=NULL;

}

return num;

}

void main()

{

clrscr(); stack obj; char ch='y';

int option,num; while(ch=='y')

{

cout<<"\n1.Push"; cout<<"\n2.Display"; cout<<"\n3.Pop"; cout<<"\n\tEnter choice";

Page no. 37

cin>>option; switch(option)

{

case 1:

cout<<"\n Enter value:"; cin>>num; obj.push(num);

cout<<"\tdo you want to continue:"; break;

case 2:

cout <<"\nStack values:"; obj.display();

cout<<"\tdo you want to continue:"; break;

case 3:

num= obj.pop();

cout<<"1 element popped out "<<num; cout<<"\tdo you want to continue:"; break;

}

ch=getch();

}

}

### Stack using Doubly Linked List

#include<conio.h> #include<stdio.h> #include<iostream.h> class stack

{

private:

public:

};

int data; stack \*next; stack \*pre;

void push(int); void display(); int pop();

stack \*top; stack \*bottom;

void stack::push(int num)

{

if((top==NULL)&&(bottom==NULL))

{

top=bottom=new stack; top->data=num;

top->next=NULL; top->pre=NULL;

}

Page no. 38

else

{

top->next=new stack; top->next->data=num; top->next->next=NULL; top->next->pre=top; top=top->next;

}

}

void stack::display()

{

stack \*q=bottom; if((top==NULL)&&(bottom==NULL))

{

cout<<"\nEmpty stack";

}

else

{

while(q!=NULL)

{

cout<<q->data<<" "; q=q->next;

}

}

}

int stack::pop()

{

int num; if((bottom==NULL)&&(top==NULL))

{

cout<<"\tEmpty stack"; return -1;

}

num=top->data; top=top->pre; if( top!=NULL)

{

delete(top->next); top->next=NULL;

}

else

{

delete(bottom); bottom=NULL;

}

return num;

}

void main()

{

clrscr(); stack s1;

Page no. 39

char ch='y';

int num,option; while(ch=='y')

{

cout<<"\n1.Add"; cout<<"\n2.Display"; cout<<"\n3.Remove"; cout<<"\n\tEnter choice: "; cin>>option; switch(option)

{

case 1:

cout<<"enter value of element: "; cin>>num;

s1.push(num); cout<<"\tElement added.";

cout<<"\nDo you want to continue : "; break;

case 2:

cout<<"\nStack Elements: "; s1.display();

cout<<"\nDo you want to continue : "; break;

case 3:

num=s1.pop();

cout<<num << " Removed"; cout<<"\nDo you want to continue : "; break;

}

ch=getch();

}

}

### Stack implementaion to check balancing of paranthesis

#include<iostream.h> #include<string.h> #include<conio.h>

class node { public:

char data; node\* next;

};

class stack {

Page no. 40

public:

node\* top;

stack() {

top = NULL;

}

void push(char x) { node\* p = new node; p->data = x;

p->next = NULL; if (top == NULL) {

top = p;

} else {

node\* save = top; top = p;

p->next = save;

}

}

char pop() {

if (top == NULL) {

cout << "UNDERFLOW";

return '\0'; // Return null character in case of underflow

} else {

node\* ptr = top;

char data = top->data; top = top->next; delete ptr;

return data;

}

}

};

int main() { clrscr();

Page no. 41

int i; stack s;

char c[30], a, y, z;

cout << "ENTER THE EXPRESSION:\n";

cin.getline(c, sizeof(c));

for (i = 0; i < strlen(c); i++) {

if ((c[i] == '(') || (c[i] == '{') || (c[i] == '[')) {

s.push(c[i]);

} else {

switch (c[i]) {

case ')':

a = s.pop();

if ((a == '{') || (a == '[')) {

cout << " INVALID EXPRESSION!! ";

getch(); return 0;

}

break; case '}':

y = s.pop();

if ((y == '[') || (y == '(')) {

cout << "INVALID EXPRESSION!!";

getch(); return 0;

}

break; case ']':

z = s.pop();

if ((z == '{') || (z == '(')) {

cout << "INVALID EXPRESSION!!";

getch(); return 0;

}

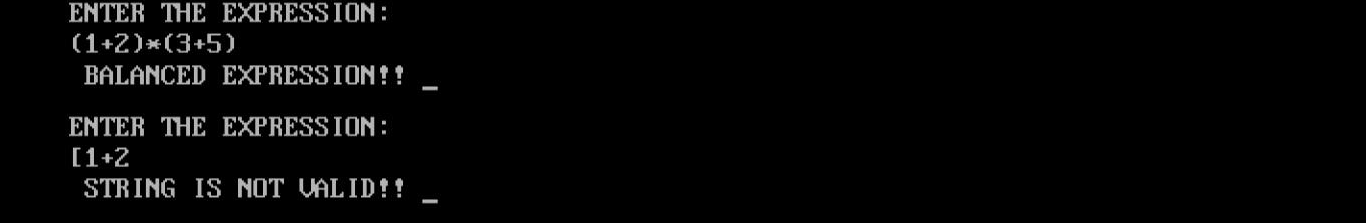
break;

}

}

Page no. 42

}



if (s.top == NULL) {

cout << " BALANCED EXPRESSION!! ";

} else {

cout << " STRING IS NOT VALID!! ";

}

getch(); return 0;

}

### Evaluate the Postfix Expression using stack

#include&lt;iostream&gt; #include&lt;ctype.h&gt; #include&lt;string.h&gt; #include&lt;conio.h&gt; using namespace std; class node

{

public:

int data; node \*next;

};

class stack

{

node \*top; public: stack()

{

top=NULL;

}

void push(int x)

{

Page no. 43

node \*p = new node(); p-&gt;data = x;

p-&gt;next = NULL; if(top==NULL)

{

top = p;

}

else

{

node \*save = top; top = p;

p-&gt;next= save;

} }

int pop()

{

if(top==NULL)

{

cout&lt;&lt;&quot;\n UNDERFLOW&quot;;

}

else

{

node \*ptr = top; top = top-&gt;next;

return(ptr-&gt;data); delete ptr;

} }

};

int main()

{

char x[30]; int a,b; stack s;

cout&lt;&lt;&quot;ENTER THE BALANCED EXPRESSION: &quot;; cin&gt;&gt;x;

for(int i= 0; i&lt; strlen(x); i++)

{

if(x[i]==&#39;\n&#39;|| x[i]==&#39;\t&#39;)

{

continue;

}

Page no. 44

if(isdigit(x[i]))

s.push(x[i]-&#39;0&#39;); else

{

a= s.pop();

b= s.pop();

switch(x[i])

{

case&#39;+&#39;:

s.push(a+b); break;

case&#39;-&#39;:

s.push(a-b); break; case&#39;\*&#39;:

s.push(a\*b); break; case&#39;/&#39;:

s.push(a/b); break;

} } }

cout&lt;&lt;&quot;\n ANSWER IS: &quot;&lt;&lt;s.pop() &lt;&lt;endl; return 0;

}

Page no. 45

## Queue:

### Simple Queue

#include<iostream.h> #include<conio.h>

class Queue{

private:

public:

};

Queue \*f; Queue \*r;

int data; Queue \*next;

void add(int); void display(); int remove();

void Queue::add(int num){

if((f==NULL) && (r==NULL))

{

}

else

{

f = r = new Queue; f->data = num;

f->next = NULL;

while(r->next!=NULL)

{

r = r->next;

}

r->next = new Queue;

//q = q->next;

r->next->data = num; r->next->next = NULL; r = r->next;

}

}

void Queue::display()

{

Queue \*q = f;

if((f==NULL) && (r==NULL)){

cout<<"Nothing to display"<<endl;

}

else{

Page no. 46

while(q!=NULL){

cout<<q->data<<" "; q = q->next;

}

}

cout<<endl;

}

int Queue::remove()

{

int num; Queue \*temp;

if((f==NULL) && (r==NULL)){

return -1;

}

else

{

}

}

temp = f;

num = f->data; f = f->next; delete(temp); return num;

void main(){

int num, opt, pos; char ch = 'y';

f = NULL; r = NULL;

Queue q; clrscr();

while(ch=='y'){

cout<<"Choose any option: "; cout<<"\n1. Add"; cout<<"\n2. Display"; cout<<"\n3. Remove"; cout<<”/n” + endl;

cin>>opt; switch(opt)

{

case 1:

cout<<"Enter your data(integer): "; cin>>num;

q.add(num); break;

Page no. 47

case 2:

cout<<"Elements are: "; q.display();

break;

case 3:

num = q.remove(); if(num == -1)

cout<<"No Queue present\n";

else break;

cout<<"Removed element: "<<num<<endl;

default:

{

}

cout<<"Enter correct choice!"<<endl; break;

}

cout<<"DO YOU WANT TO CONTINUE? (y/n): "<<endl;

ch = getch();

}

}

### Doubly Ended Queue

#include<conio.h> #include<stdio.h> #include<iostream.h> class queue

{

private:

public:

};

int data; queue \*next; queue \*prv;

void add(int); void display(); int remove(); void addf(int);

int remover(void);

queue \*front; queue \*rear;

void queue::add(int num)

{

Page no. 48

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

{

front=rear=new queue; front->data=num;

front->next=NULL; front->prv=NULL;

}

else

{

rear->next=new queue; rear->next->data=num; rear->next->next=NULL; rear->next->prv=rear; rear=rear->next;

}

}

void queue::display()

{

queue \*q=front; if((front==NULL)&&(rear==NULL))

{

cout<<"\nStack empty";

}else

{

while(q!=NULL)

{

cout<<q->data<<" "; q=q->next;

} } }

int queue::remove()

{

int num; queue \*temp;

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

{

return -1;

}

else

{

temp=front; num=front->data; front=front->next; front->prv=NULL; delete(temp);

}

return num;

}

void queue::addf(int num)

{

queue \*temp; if((front==NULL)&&(rear==NULL))

Page no. 49

{

front=rear=new queue; front->data=num;

front->next=NULL; front->prv=NULL;

}

else{ temp=front;

front=new queue; front->data=num; front->prv=NULL; front->next=temp; front->next->prv=front;

}

}

int queue::remover(void)

{

queue \*temp; int num;

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

{

return -1;

}else

{

temp=rear; num=rear->data; rear=rear->prv; rear->next=NULL; delete(temp); return num;

}}

void main()

{

queue z; clrscr(); char c='y'; int num,op; while(c=='y')

{

cout<<"\n1.Add"; cout<<"\n2.Display"; cout<<"\n3.Remove"; cout<<"\n4.Add from Front"; cout<<"\n5. Remove from Rear"; cout<<"\n Enter an option"; cin>>op;

switch(op)

{case 1:

cout<<"\nEnter value: "; cin>>num;

z.add(num);

Page no. 50

cout<<"\n\tDo you want to continue: "; break;

case 2:

cout<<"Queue values:\t"; z.display();

cout<<"\n\tDo you want to continue: "; break;

case 3:

num=z.remove();

cout<< num <<" removed";

cout<< "Do you want to continue "; break;

case 4:

cout<<"Enter number"; cin>>num; z.addf(num);

cout<<"Do you want to continue"; break;

case 5:

num=z.remover();

cout<< num <<" removed";

cout<< "Do you want to continue "; break;

}

c=getch();

}

}

### Circular Queue

#include<iostream.h> #include<conio.h>

class Queue { private:

int data; Queue\* next;

public:

void add(int); void display(); int remove();

};

Queue\* front; Queue\* rear;

void Queue::add(int num) { Queue\* newNode = new Queue; newNode->data = num; newNode->next = NULL;

Page no. 51

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

rear->next = front;

}

else {

rear->next = newNode; rear = newNode;

rear->next = front;

}

}

void Queue::display() {

if (front == NULL && rear == NULL) { cout << "Nothing to display" << endl;

}

else {

Queue\* temp = front; do {

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

} while (temp != front);

}

cout << endl;

}

int Queue::remove() { int num;

if (front == NULL && rear == NULL) { return -1;

}

else {

Queue\* temp = front; num = front->data;

if (front == rear) {

front = rear = NULL;

}

else {

front = front->next; rear->next = front;

}

delete(temp); return num;

}

}

void main() { int num, opt;

Page no. 52

char ch = 'y'; front = NULL; rear = NULL; Queue q;

clrscr();

while (ch == 'y') {

cout << "Choose any option: "; cout << "\n1. Add";

cout << "\n2. Display"; cout << "\n3. Remove"; cout << endl;

cin >> opt;

switch (opt) { case 1:

cout << "Enter your data (integer): "; cin >> num;

q.add(num); break;

case 2:

cout << "Elements are: "; q.display();

break;

case 3:

num = q.remove(); if (num == -1)

cout << "No Queue present\n"; else

cout << "Removed element: " << num << endl; break;

default: {

cout << "Enter the correct choice!" << endl; break;

}

}

cout << "DO YOU WANT TO CONTINUE? (y/n): ";

ch = getch();

}

}

Page no. 53

### Priority Queue

#include<conio.h> #include<stdio.h> #include<iostream.h> class queue

{ private:

public:

};

int data; queue \*next;

void add(int); void display(); int remove();

void priority(void);

queue \*front; queue \*rear;

void queue::add(int num)

{

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

{

front=rear=new queue; front->data=num;

front->next=NULL;

}

else

{

rear->next=new queue; rear->next->data=num; rear->next->next=NULL; rear=rear->next;

}

}

void queue::display()

{

queue \*q=front; if((front==NULL)&&(rear==NULL))

{

cout<<"\nStack empty";

}

else

{

while(q!=NULL)

{

cout<<q->data<<" "; q=q->next;

}

}

}

int queue::remove()

{

Page no. 54

int num; queue \*temp;

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

{

return -1;

}

else

{

temp=front; num=front->data; front=front->next; delete(temp);

}

return num;

}

void queue::priority(void)

{

int num; if((front==NULL)&&(rear==NULL))

{

cout<<"Queue is Empty";

}

else

{

if(front->data>rear->data)

{

num=remove(); cout<<num;

}

else

{

cout<<"Enter Number"; cin>>num;

add(num);

}

}

}

void main()

{

queue z; clrscr(); char c='y'; int num,op; while(c=='y')

{

cout<<"\n1.Add"; cout<<"\n2.Display"; cout<<"\n3.Remove"; cout<<"\n4.Priority"; cout<<"\n Enter an Option"; cin>>op;

Page no. 55

switch(op)

{

case 1:

cout<<"\nEnter value: "; cin>>num;

z.add(num);

cout<<"\n\tDo you want to continue: "; break;

case 2:

cout<<"Queue values:\t"; z.display();

cout<<"\n\tDo you want to continue: "; break;

case 3:

num=z.remove();

cout<< num <<" removed";

cout<< "Do you want to continue "; break;

case 4:

z.priority();

cout<<"Do you want to continue"; break;

}

c=getch();

}

}

Page no. 56

## Binary Search Tree:

#include<iostream.h> #include<conio.h>

class bst{

private:

public:

int data; bst \*left; bst \*right;

void add(int,bst\*\*); void preorder(bst\*); void postorder(bst\*); void inorder(bst\*); void max(bst \*q); void min(bst \*q); void count(bst \*\*); int search (int,bst \*\*);

};

bst \*root;

void bst ::add(int num,bst \*\*q){ if(\*q==NULL){

(\*q)=new bst; (\*q)->data=num;

(\*q)->left=NULL; (\*q)->right=NULL;

}

else{

if(num<(\*q)->data){

add(num,&((\*q)->left));

}

else{

}

}

}

add(num,&((\*q)->right));

void bst :: preorder(bst \*q){

if(q!=NULL){

cout<<q->data<<" "; preorder(q->left); preorder(q->right);

}

}

void bst :: postorder(bst \*q){

if(q!=NULL){

postorder(q->left); postorder(q->right); cout<<q->data<<" " ;

Page no. 57

}

}

void bst :: inorder(bst \*q){

if(q!=NULL){

inorder(q->left); cout<<q->data<<" " ; inorder(q->right);

}

}

void bst::max(bst \*q){

if(q==NULL)

cout<<"Tree doesnt exit\n";

else{

while(q->right != NULL)

{

q = q->right;

}

cout<<q->data<<endl;

}

}

void bst::min(bst \*q){

if(q==NULL)

cout<<"Tree doesnt exit\n";

else{

while(q->left != NULL)

{

q = q->left;

}

cout<<q->data<<endl;

}

}

void bst::count(bst \*\*q)

{

int c=0;

bst \*temp=\*q; if(root==NULL)

{

}

else

{

cout<<c;

c=1;

while(temp->left!=NULL)

{

c++;

Page no. 58

temp=temp->left;

}

temp=(\*q);

while(temp->right!=NULL)

{

c++;

temp=temp->right;

}

cout<<"Number of Nodes"<<c;

}

}

void main()

{

clrscr();

int num,option; char ch='y'; root=NULL; bst b; while(ch=='y'){

cout<<"\n1.add:"; cout<<"\n2.preorder:"; cout<<"\n3. postorder"; cout<<"\n4. in order"; cout<<"\n5. Max element"; cout<<"\n6. Min element"; cout<<"\n7. Count elements"; cout<<"\n7. Search elements"; cout<<"\n Enter an option:";

cout<<"\n"; cin>>option;

switch(option)

{

case 1:

case 2:

case 3:

cout<<"Enter number:"; cin>>num; b.add(num,&root); cout<<" ";

break;

b.preorder(root); cout<<" "; break;

b.postorder(root);

Page no. 59

case 4:

cout<<" "; break;

b.inorder(root); break;

case 5:

b.max(root); break;

case 6:

b.min(root); break;

case 7:

b.count(&root); break;

case 8:

cout<<"\n enter the number"; cin>>num;

pos = b.search(num,&root); if(pos==-1)

{

}

else

{

}

cout<<"\n number not found";

cout<<"\n number found";

cout<<"\n do you want to continue "; break;

}

cout<<"Do you want to continue"; ch=getch();

}

}

## Heap:

#include<iostream> #include<vector>

using namespace std;

class MinHeap { private:

Page no. 60

vector<int> heap;

// Helper function to heapify a subtree with the root at given index void heapify(int index) {

int leftChild = 2 \* index + 1; int rightChild = 2 \* index + 2; int smallest = index;

// Compare with left child

if (leftChild < heap.size() && heap[leftChild] < heap[smallest]) { smallest = leftChild;

}

// Compare with right child

if (rightChild < heap.size() && heap[rightChild] < heap[smallest]) { smallest = rightChild;

}

// If the smallest is not the root, swap and recursively heapify the affected subtree if (smallest != index) {

swap(heap[index], heap[smallest]); heapify(smallest);

}

}

public:

// Function to build a heap from a given array void buildHeap(const vector<int>& arr) {

heap = arr;

// Start from the last non-leaf node and heapify all nodes in reverse order for (int i = (heap.size() / 2) - 1; i >= 0; i--) {

heapify(i);

}

}

// Function to display the elements of the heap void displayHeap() {

cout << "Heap: "; for (int num : heap) {

Page no. 61

cout << num << " ";

}

cout << endl;

}

// Function to delete the minimum element from the heap (root) void deleteMin() {

if (heap.empty()) {

cout << "Heap is empty. Cannot delete." << endl; return;

}

// Replace the root with the last element heap[0] = heap.back(); heap.pop\_back();

// Heapify the root heapify(0);

}

};

int main() {

MinHeap minHeap;

// Example usage

vector<int> inputArray = {4, 10, 3, 5, 1};

minHeap.buildHeap(inputArray); minHeap.displayHeap();

// Deleting the minimum element minHeap.deleteMin(); minHeap.displayHeap();

return 0;

}

Page no. 62

## Graph:

### Adjacency Matrix - Directed

#include<iostream.h> #include<conio.h>

class DAdjMatrix

{

private: public:

int adjMatrix[20][20], n;

void createGraph(); void display();

void insertV(); void deleteV();

void insertE(int, int); void deleteE(int ,int);

DAdjMatrix()

{

for(int i=1; i<=20; i++){

for(int j=1; j<=20; j++){

adjMatrix[i][j] = 0;

}

}

}

};

void DAdjMatrix :: createGraph()

{

int i, maxEdge, or, ds;

cout<<"Enter no of vertices: "; cin>>n;

cout<<endl;

maxEdge = (n\*(n-1)) / 2; cout<<"Enter edges:\n";

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

{

cout<<"Enter origin=0 and des=0 to exit\n";

cout<<"Enter origin: "; cin>>or;

Page no. 63

cout<<"Enter des: "; cin>>ds;

if(or==0 || ds==0)

break;

if(or>n || or<0 || ds>n || ds<0)

{

cout<<"Invalid Edge\n"; i--;

continue;

}

adjMatrix[or][ds] = 1;

}

}

void DAdjMatrix :: display()

{

int i, j;

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

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

cout<<adjMatrix[i][j]<<"\t";

}

cout<<"\n";

}

}

void DAdjMatrix::insertV()

{

int i; n++;

cout<<"\nNo. of vertices: "<<n<<endl;

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

{

adjMatrix[i][n] = 0;

adjMatrix[n][i] = 0;

}

}

void DAdjMatrix::deleteV()

{

int i;

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

{

adjMatrix[i][n] = 0;

adjMatrix[n][i] = 0;

}

n--;

cout<<"\nNo. of vertices: "<<n<<endl;

}

void DAdjMatrix::insertE(int or, int ds)

Page no. 64

{

if(or>n || or<0 || ds>n || ds<0)

{

}

else{

}

}

cout<<"Invalid Edge\n";

adjMatrix[or][ds] = 1;

void DAdjMatrix::deleteE(int or, int ds)

{

if(or>n || or<0 || ds>n || ds<0)

{

}

void main()

{

}

else{

}

cout<<"Invalid Edge\n";

adjMatrix[or][ds] = 0;

DAdjMatrix m; int opt, or, ds; char ch='y';

clrscr(); m.createGraph(); while(ch=='y'){

cout<<"Choose any option: ";

cout<<"\n0. Display"; cout<<"\n1. Insert Vertex"; cout<<"\n2. Delete Vertex"; cout<<"\n3. Insert Edge"; cout<<"\n4. Delete Edge";

cout<<endl; cin>>opt;

switch(opt)

{

case 0:

m.display(); break;

case 1:

Page no. 65

m.insertV(); break;

case 2:

m.deleteV(); break;

case 3:

cout<<"Enter origin: "; cin>>or;

cout<<"Enter des: "; cin>>ds;

m.insertE(or, ds); break;

case 4:

cout<<"Enter origin: "; cin>>or;

cout<<"Enter des: "; cin>>ds;

m.deleteE(or, ds); break;

default:

{

}

cout<<"Enter correct choice!"<<endl; break;

}

cout<<"DO YOU WANT TO CONTINUE? (y/n): "<<endl;

ch = getch();

}

}

Page no. 66

### Adjacency Matrix - Undirected

#include<iostream.h> #include<conio.h>

class UAdjMatrix

{

private: public:

int adjMatrix[20][20], n;

void createGraph(); void display();

void insertV(); void deleteV();

void insertE(int, int); void deleteE(int ,int);

UAdjMatrix()

{

for(int i=1; i<=20; i++){

for(int j=1; j<=20; j++){

adjMatrix[i][j] = 0;

}}}};

void UAdjMatrix :: createGraph()

{

int i, maxEdge, or, ds; cout<<"Enter no of vertices: "; cin>>n;

cout<<endl;

maxEdge = (n\*(n-1)) / 2; cout<<"Enter edges:\n"; for(i=1; i<=maxEdge; i++)

{

cout<<"Enter origin=0 and des=0 to exit\n";

cout<<"Enter origin: "; cin>>or;

cout<<"Enter des: "; cin>>ds;

if(or==0 || ds==0)

break;

if(or>n || or<0 || ds>n || ds<0)

{

Page no. 67

cout<<"Invalid Edge\n"; i--;

continue;

}

adjMatrix[or][ds] = 1;

adjMatrix[ds][or] = 1;

}

}

void UAdjMatrix :: display()

{

int i, j;

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

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

cout<<adjMatrix[i][j]<<"\t";

}

cout<<"\n";

}

}

void UAdjMatrix::insertV()

{

int i; n++;

cout<<"\nNo. of vertices: "<<n<<endl;

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

{

adjMatrix[i][n] = 0;

adjMatrix[n][i] = 0;

}

}

void UAdjMatrix::deleteV()

{

int i;

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

{

adjMatrix[i][n] = 0;

adjMatrix[n][i] = 0;

}

n--;

cout<<"\nNo. of vertices: "<<n<<endl;

}

void UAdjMatrix::insertE(int or, int ds)

{

if(or>n || or<0 || ds>n || ds<0)

{

cout<<"Invalid Edge\n";

Page no. 68

}

else{

}

}

adjMatrix[or][ds] = 1;

adjMatrix[ds][or] = 1;

void UAdjMatrix::deleteE(int or, int ds)

{

if(or>n || or<0 || ds>n || ds<0)

{

}

void main()

{

}

else{

}

cout<<"Invalid Edge\n";

adjMatrix[or][ds] = 0;

adjMatrix[ds][or] = 0;

UAdjMatrix m; int opt, or, ds; char ch='y';

clrscr(); m.createGraph();

while(ch=='y'){

cout<<"Choose any option: "; cout<<"\n0. Display"; cout<<"\n1. Insert Vertex"; cout<<"\n2. Delete Vertex"; cout<<"\n3. Insert Edge"; cout<<"\n4. Delete Edge";

cout<<endl;

cin>>opt; switch(opt)

{

case 0:

m.display(); break;

case 1:

m.insertV(); break;

Page no. 69

case 2:

m.deleteV(); break;

case 3:

cout<<"Enter origin: "; cin>>or;

cout<<"Enter des: "; cin>>ds;

m.insertE(or, ds); break;

case 4:

cout<<"Enter origin: "; cin>>or;

cout<<"Enter des: "; cin>>ds;

m.deleteE(or, ds); break;

default:

{

}

cout<<"Enter correct choice!"<<endl; break;

}

cout<<"DO YOU WANT TO CONTINUE? (y/n): "<<endl;

ch = getch();

}

}

Page no. 70

#### Graph traversal

#include<iostream.h> #include<conio.h>

class gdfs

{

private:

int adj[10][10]; int visitedarr[10]; int n;

public:

void buildadj(); void dfs(int);

gdfs()

{

int i,j; for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

adj[i][j]=0;

}

visitedarr[i]=0;

}

}

};

void gdfs :: buildadj()

{

int i,j;

cout<<"Enter number of vertices: "; cin>>n;

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

{

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

{

cin>>adj[i][j];

}

}

};

void gdfs :: dfs(int x)

{

int j; visitedarr[x] = 1;

cout<< x << " is visited."<< endl;

Page no. 71

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

{

if((adj[x][j] == 1)&&(visitedarr[j] == 0))

{

dfs(j);

}

}

};

void main()

{

clrscr(); gdfs d; d.buildadj();

d.dfs(0);

getch();

}

Page no. 72

1. Project

# Code:

#include<stdio.h> #include<iostream> #include<math.h> using namespace std;

//Abhijeet Ramdas Khadsare

// Define structures for the three stacks struct node1

{

int data1; node1 \*next1;

}\*top1 = NULL, \*p1 = NULL, \*np1 = NULL;

struct node2

{

int data2; node2 \*next2;

}\*top2 = NULL, \*p2 = NULL, \*np2 = NULL;

struct node3

{

int data3; node3 \*next3;

}\*top3 = NULL, \*p3 = NULL, \*np3 = NULL;

#### TOWER OF HANOI

// Function to push an element onto the first stack void push1(int data)

{

np1 = new node1; np1->data1 = data; np1->next1 = NULL;

Page no. 73

if (top1 == NULL)

{

top1 = np1;

}

else

{

np1->next1 = top1; top1 = np1;

}

} // Function to pop an element from the first stack int pop1()

{

int b = 999;

if (top1 == NULL)

{

return b;

}

else

{

p1 = top1;

top1 = top1->next1; return(p1->data1); delete(p1);

}

}

void push2(int data)

{

np2 = new node2; np2->data2 = data; np2->next2 = NULL; if (top2 == NULL)

Page no. 74

{

top2 = np2;

}

else

{

np2->next2 = top2; top2 = np2;

}

}

int pop2()

{

int b = 999;

if (top2 == NULL)

{

return b;

}

else

{

p2 = top2;

top2 = top2->next2; return(p2->data2); delete(p2);

}

}

void push3(int data)

{

np3 = new node3; np3->data3 = data; np3->next3 = NULL; if (top3 == NULL)

{

Page no. 75

top3 = np3;

}

else

{

np3->next3 = top3; top3 = np3;

}

}

int pop3()

{

int b = 999;

if (top3 == NULL)

{

return b;

}

else

{

p3 = top3;

top3 = top3->next3; return(p3->data3); delete(p3);

}

} // Function to get the top of the smallest non-empty stack int top\_of\_stack()

{

if (top1 != NULL && top1->data1 == 1 )

{

return 1;

}

else if (top2 != NULL && top2->data2 == 1)

{

Page no. 76

return 2;

}

else if (top3 != NULL && top3->data3 == 1)

{

return 3;

}

}

// Display functions for each stack void display1()

{

cout<<endl; node1 \*p1; p1 = top1;

cout<<"Tower1-> "<<"t"; while (p1 != NULL)

{

cout<<p1->data1<<"t"; p1 = p1->next1;

}

cout<<endl;

}

void display2()

{

node2 \*p2; p2 = top2;

cout<<"Tower2-> "<<"t"; while (p2 != NULL)

{

cout<<p2->data2<<"t"; p2 = p2->next2;

}

Page no. 77

cout<<endl;

}

void display3()

{

node3 \*p3; p3 = top3;

cout<<"Tower3-> "<<"t"; while (p3 != NULL)

{

cout<<p3->data3<<"t"; p3 = p3->next3;

}

cout<<endl; cout<<endl;

} // Tower of Hanoi function(toh- tower of hanoi) void toh(int n)

{

int i, x, a, b;

// Loop through the moves of the Tower of Hanoi for (i = 0; i < (pow(2,n)); i++)

{

display1(); display2(); display3();

x = top\_of\_stack(); if (i % 2 == 0)

{

if (x == 1)

{

push3(pop1());

}

Page no. 78

else if (x == 2)

{

push1(pop2());

}

else if (x == 3)

{

push2(pop3());

}

}

else

{

if (x == 1)

{

a = pop2();

b = pop3();

if (a < b && b != 999)

{

push3(b);

push3(a);

}

else if (a > b && a != 999)

{

push2(a);

push2(b);

}

else if (b == 999)

{

push3(a);

}

else if (a == 999)

{

Page no. 79

push2(b);

}

}

else if (x == 2)

{

a = pop1();

b = pop3();

if (a < b && b != 999)

{

push3(b);

push3(a);

}

else if (a > b && a != 999)

{

push1(a);

push1(b);

}

else if (b == 999)

{

push3(a);

}

else if (a == 999)

{

push1(b);

}

}

else if (x == 3)

{

a = pop1();

b = pop2();

if (a < b && b != 999)

Page no. 80

{

push2(b);

push2(a);

}

else if (a > b && a != 999)

{

push1(a);

push1(b);

}

else if (b == 999)

{

push2(a);

}

else if (a == 999)

{

push1(b);

} // Handling moves for odd steps in Tower of Hanoi

}

}

}

}

// Main function int main()

{

int n, i;

cout<<"enter the number of disks: "; cin>>n;

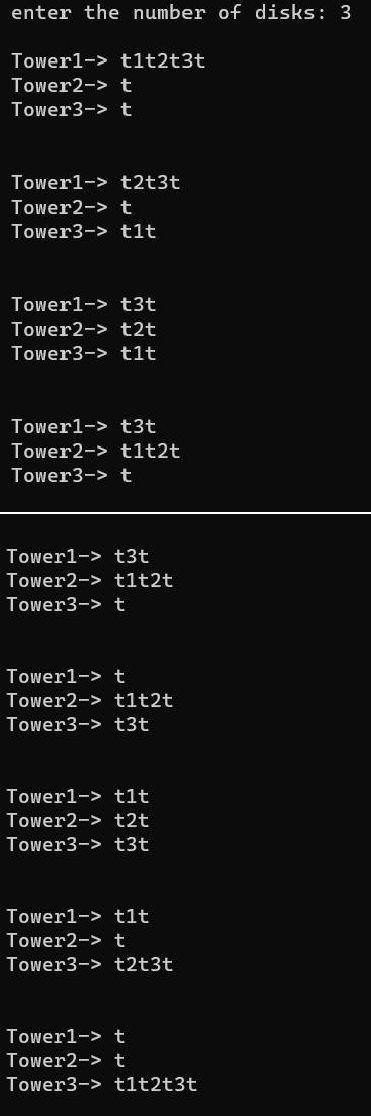
// Initialize the first tower with disks for (i = n; i >= 1; i--)

{

push1(i);

Page no. 81

} // Solve the Tower of Hanoi toh(n);



return 0;

}

#### Output:

Page no. 82