**Data Structures Lab – Batch B2**

|  |  |  |
| --- | --- | --- |
| ***Unit 1*** | ***Sorting*** | ***Date*** |
| 1. | Bubble Sort | 01.09.2023 |
| 2. | Selection Sort | 04.09.2023 |
| 3. | Insertion Sort | 09.09.2023 |
| 4. | Radix Sort | 11.09.2023 |
| 5. | Shell Sort | 16.09.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 2*** | ***Searching*** | ***Date*** |
| 6. | Linear Search | 25.08.2023 |
| 7. | Binary Search | 28.08.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 2*** | ***Hashing*** | ***Date*** |
| 8. | Modulo Division with Linear Probe | 11.12.2023 |
| 9. | Digit Extraction with Linear Probe |  |
| 10. | Fold Shift with Linear Probe |  |
| 11. | Fold Boundary with Linear Probe |  |

|  |  |  |
| --- | --- | --- |
| ***Unit 3*** | ***Stacks*** | ***Date*** |
| 12. | Array Based Stack | 30.10.2023 |
| 13. | List Based Stack | 04.11.2023 |
| 14. | Evaluation of Postfix Expression using Stacks | 06.11.2023 |
| 15. | Balancing of Parenthesis using Stacks | 09.11.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 4*** | ***Queues*** | ***Date*** |
| 16. | Ordinary Queue | 09.11.2023 |
| 17. | Circular Queue | 25.11.2023 |
| 18. | Priority Queue | 04.12.2023 |
| 19. | Double-Ended Queue | 02.12.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 5*** | ***Linked Lists*** | ***Date*** |
| 20. | Singly Linked List | 18.09.2023 30.09.2023 |
| 21. | Circular Linked List | 09.10.2023 |
| 22. | Doubly Linked List | 14.10.2023  16.10.2023 |
| 23. | Polynomial Addition using Lists | 21.10.2023  28.10.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 6*** | ***Trees*** | ***Date*** |
| 24. | Binary Search Trees | 05.12.2023 |
| 25. | Max-Heaps | 11.12.2023 |
| 26. | Min-Heaps | 11.12.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 7*** | ***Graphs*** | ***Date*** |
| 27. | Representation of Graphs using Adjacency Matrix | 07.12.2023 |
| 28. | Perform Breadth-First Traversal on a Graph using Queues | 07.12.2023 |
| 29. | Find MST using Kruskal’s Algorithm | 09.12.2023 |

|  |  |
| --- | --- |
| **Program No:** | **1** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 1: Sorting – Bubble Sort** |
| **Objective :** | **Bubble Sort** |

**Source Code:**

/\*

Name:Viraj Ghogale

Roll No:1418

Unit 1:Sorting

Program:Bubble sort \*/

#include<iostream>

using namespace std;

int main()

{

int A[20],n;

int i,j;

int tmp;

cout<<"\*\*\* Bubble Sort \*\*\*"<<endl<<endl;

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

cin>>n;

cout<<" Enter "<<n<<" elements: "<<endl;

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

{

cin>>A[i];

}

//Bubble Sort

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

{

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

{

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

{

tmp=A[i];

A[i]=A[j];

A[j]=tmp;

}

}//end of for j

} //end of for i

cout<<"After Bubble Sort:";

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

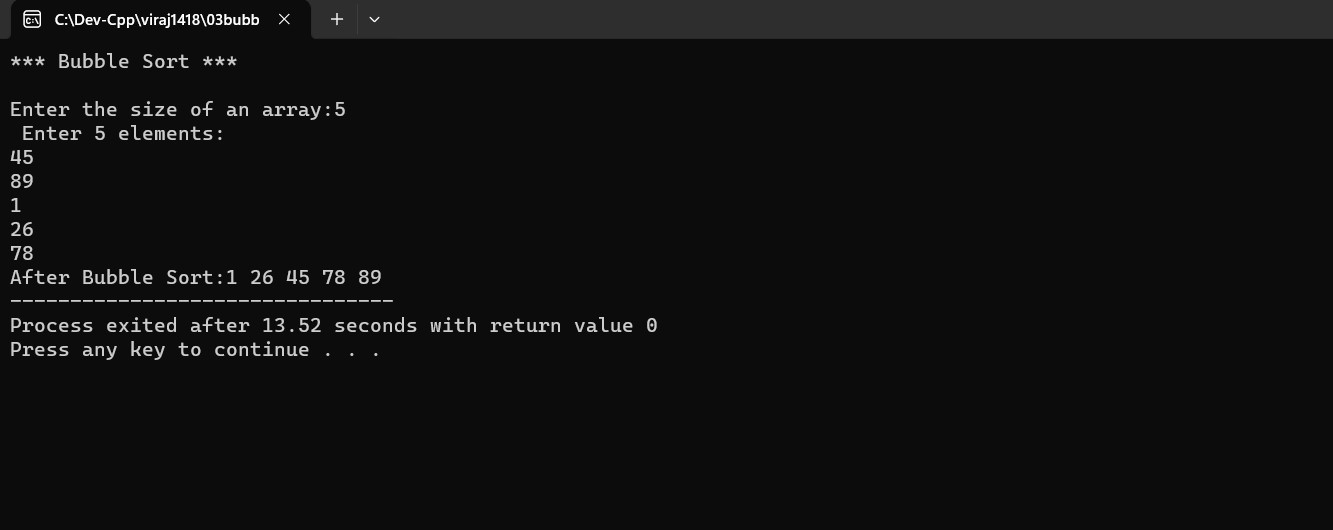
{

cout<<A[i] <<" ";

}

}//end of main

**Output:**



|  |  |
| --- | --- |
| **Program No:** | **2** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 1: Sorting – Selection Sort** |
| **Objective :** | **Selection Sort** |

**Source Code:**

/\* Viraj Ghogale

FY-MCA

1418

Unit 1: Sorting

Program: Selection Sort

14/09/2023

\*/

#include <iostream>

using namespace std;

int main() { int A[20], n, i, j; int min,loc,tmp;

cout << "------ SELECTION SORT ------"<<endl; cout << "Enter the size of the array: "; cin >> n;

cout << "Enter " << n <<" values in the array: " <<endl<<endl;

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

{ cin >> A[i];

}

//SELECTION SORT

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

{

//ASSUME MIN NUMBER TO FIRST ELEMRNT

min = A[i];

loc = i;

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

{

if (A[j] < min) { min = A[j]; loc = j; }

}

tmp = A[i]; A[i] = A[loc];

A[loc] = tmp;

}

cout << "Sorted Arrays: "; for(i=0; i<n; i++)

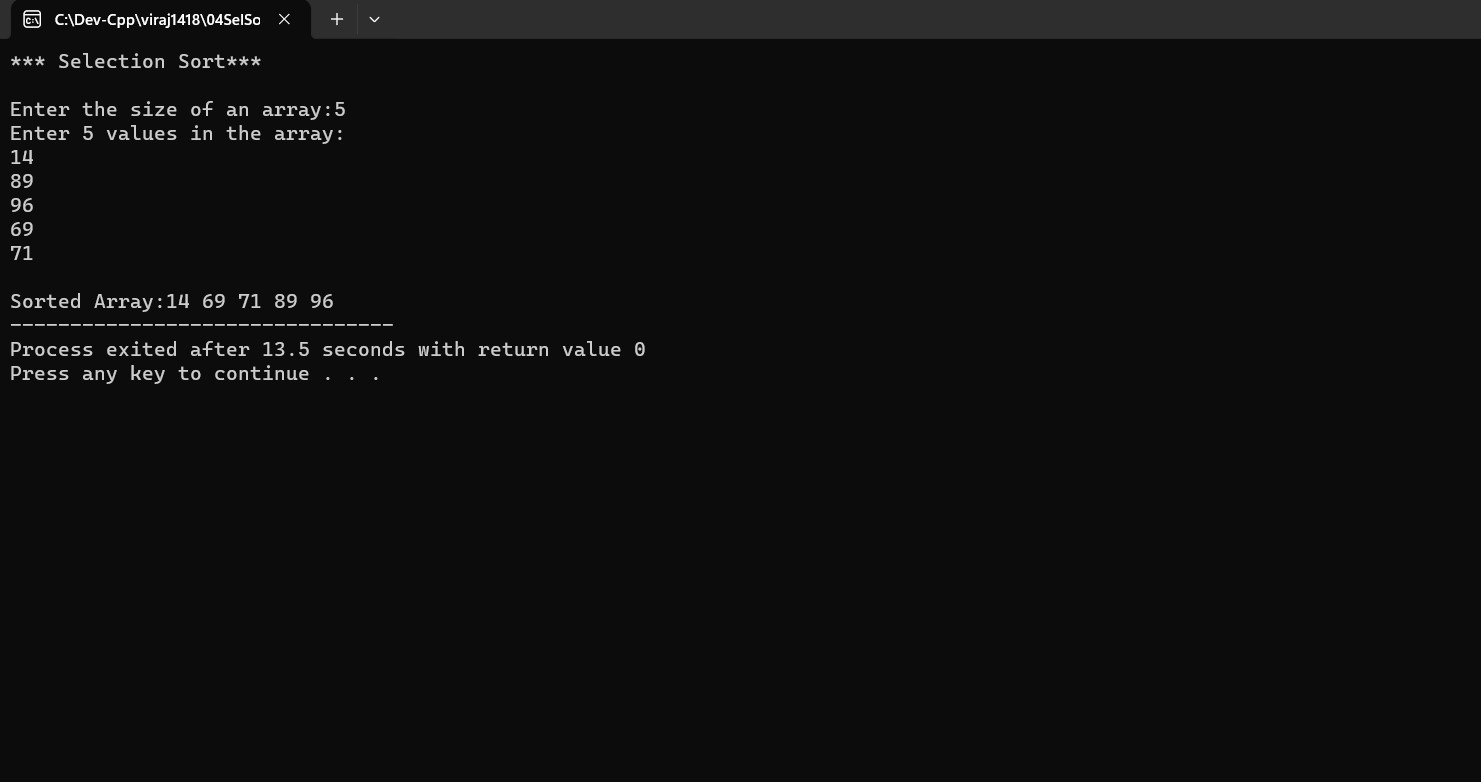
{

cout << A[i] << " ";

}

} //END OF MAIN

**Output:**



|  |  |
| --- | --- |
| **Program No:** | **3** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 1: Sorting – Insertion Sort** |
| **Objective :** | **Insertion Sort** |

**Source Code:**

/\* Viraj Ghogale

FY-MCA

1418

Unit 1: Sorting

Program: Insertion Sort

14/09/2023

\*/

#include <iostream>

using namespace std;

int main() { int A[20], n, i; int tmp, loc;

cout << "------ INSERTION SORT ------"<<endl; cout << "Enter the size of the array: "; cin >> n;

cout << "Enter " << n <<" values in the array: " <<endl<<endl;

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

{ cin >> A[i];

}

//INSERTION SORT.

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

{ tmp = A[i]; loc = i-1; while(tmp<A[loc] && loc>=0)

{

A[loc+1] = A[loc];

loc--;

}

A[loc+1] = tmp;

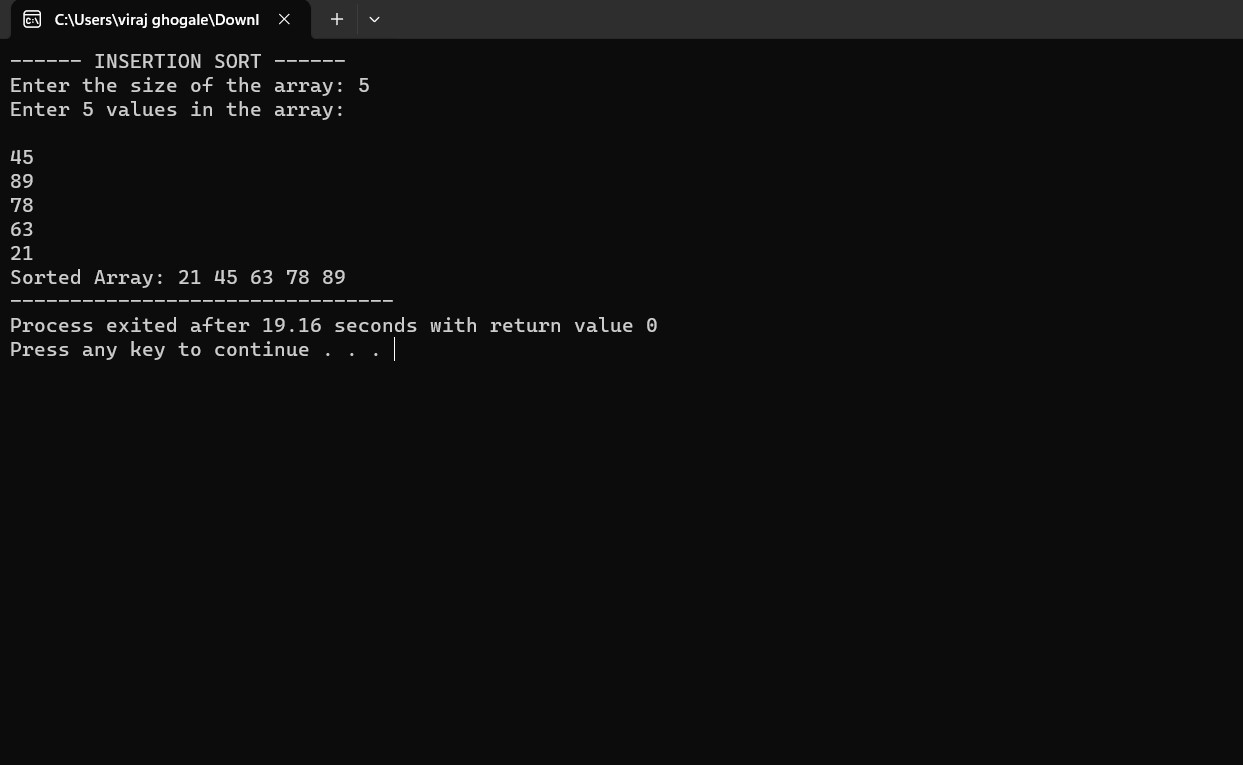
}

cout << "Sorted Array: " ; for(i=0; i<n; i++){ cout << A[i] << " ";

}

}//END OF MAIN

**Output:**



|  |  |
| --- | --- |
| **Program No:** | **4** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 1: Sorting – Radix Sort** |
| **Objective :** | **Radix Sort** |

**Source Code:**

/\* Name : Viraj Ghogale

Roll no:1418

Unit 2:Sorting

Program:Radix sort

\*/

#include<iostream>

using namespace std;

int main()

{

int a[20],n,j,max, i;

int passes=0, pass\_no, divisor=1;

int r,k, bucket[10][20], b\_count[10];

cout<< "\*\*\*Radix Sort\*\*\*\n\n";

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

cin>>n;

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

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

{

cin>>a[i];

}

//Radix sort

//Identify the maximum number in the array

max = a[0];

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

{

if (a[i] > max)

{

max =a[i];

}

}

//2. Count digits in max

while(max>0)

{

passes++;

max = max/10;

}

//3. Radix sort process

for(pass\_no=1 ; pass\_no<=passes ;pass\_no++)

{

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

{

b\_count[k] =0;

}

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

{

r = ( a[i]/divisor)%10;

bucket[r][b\_count[r]] = a[i];

b\_count[r]++;

}

//Collect bins

i=0;

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

{

for(j=0 ; j<b\_count[k] ; j++)

{

a[i] = bucket[k][j];

i++;

}

}//outer for k

divisor = divisor \* 10;

}//end for pass\_no

cout <<"Sorted Array:";

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

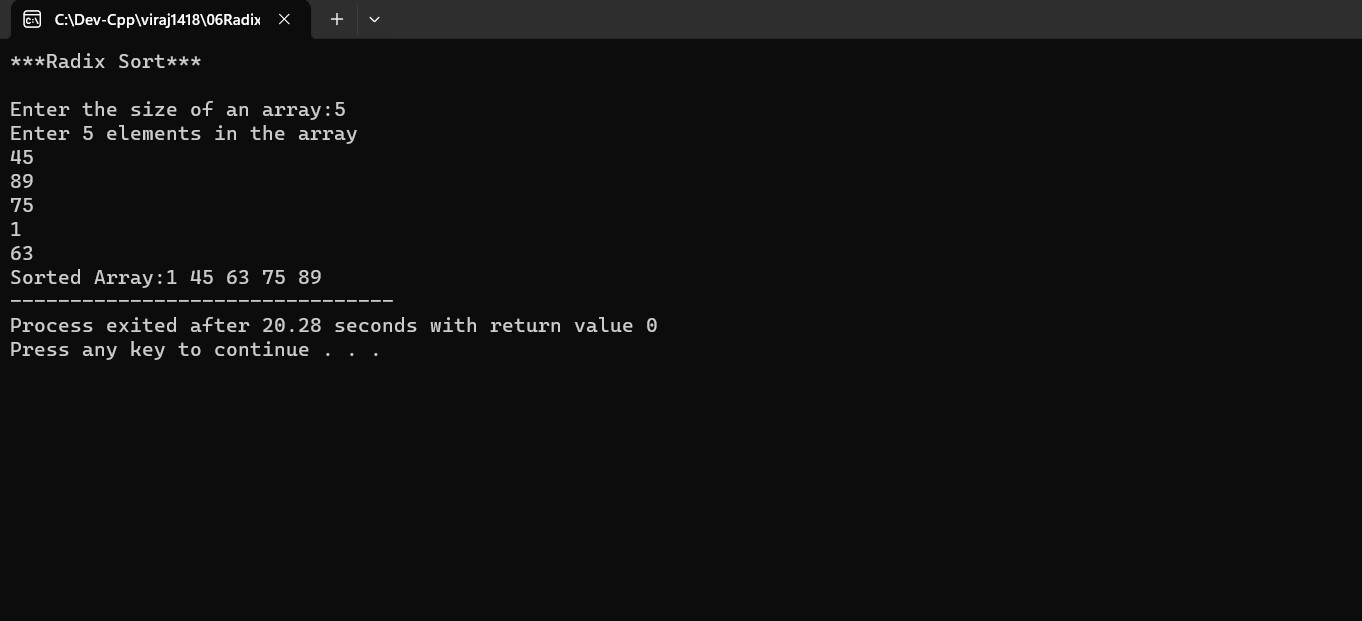
{

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

}

}// end of main

**Output:**



|  |  |
| --- | --- |
| **Program No:** | **5** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 1: Sorting – Shell Sort** |
| **Objective :** | **Shell Sort** |

**Source Code:**

/\*Name :Viraj Ghogale

Roll No:1418

Unit:1

Program:Shell Sort

\*/

#include<iostream>

#include<math.h>

using namespace std;

int main()

{

int a[20],i,n;

int gap,pos,extractItem;

cout << "\*\*\*Shell Sort\*\*\*"<<endl<<endl;

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

cin>>n;

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

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

{

cin>>a[i];

}

//Shell Sort

gap =floor(n/2);

while(gap>0)

{

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

{

extractItem = a[i+gap];

pos = i+gap;

while((pos-gap)>=0 && extractItem<a[pos-gap])

{

a[pos]= a[pos-gap];

pos =pos-gap;

}//end for inner while loop

a[pos]= extractItem;

}//end of for i

gap=floor(gap/2);

}//end of while loop

cout << "Sorted Array: ";

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

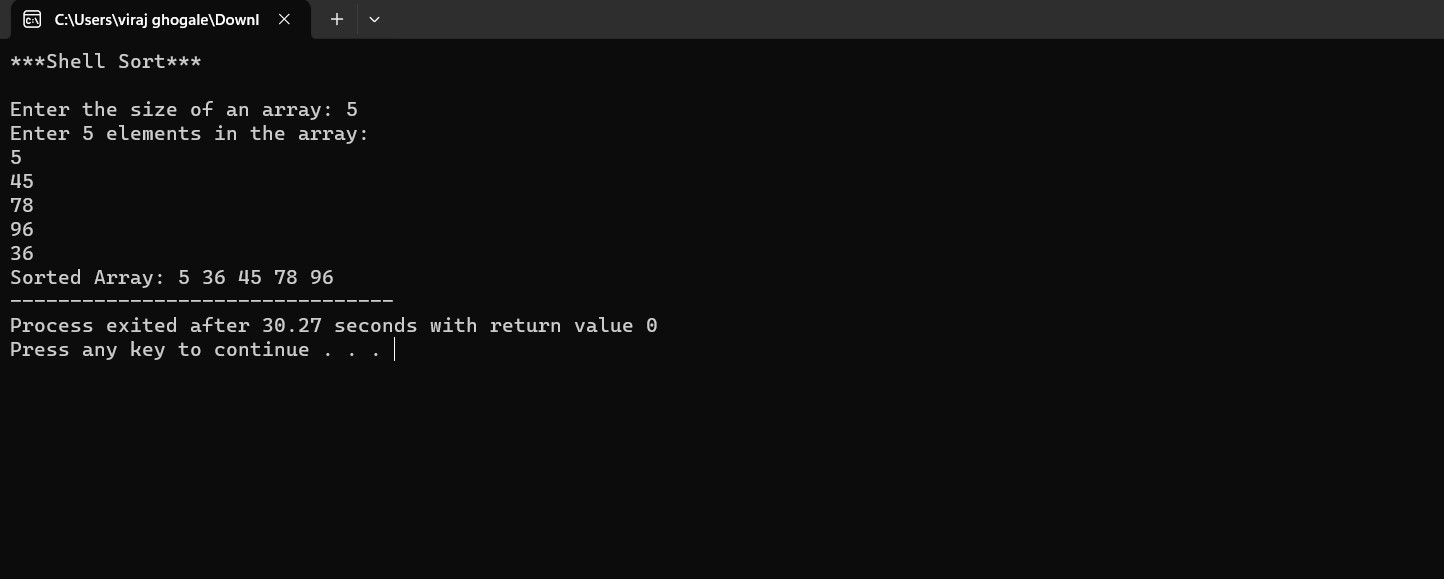
{

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

}

}//end of main

**Output:**



|  |  |
| --- | --- |
| **Program No:** | **6** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 2: Searching – Linear Search** |
| **Objective :** | **Linear Searching** |

**Source Code:**

/\* Name:Viraj Ghogale

Roll No : 1418

Unit:2

Program: Linear search

\*/

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int num [10];

int target;

int size;

int i;

int flag=0;//Flag not found

cout<<"\*\*\*Linear Search\*\*\*"<<endl<<endl;

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

cin>>size;

cout<<"Enter"<<size<<" the values in the array:"<<endl;

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

{

cin>>num[i];

}

cout<<"Enter the target value:"<<endl;

cin>>target;

//Linear search

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

{

if (num[i] ==target)

{

flag=1;//target found so raise the flag

break;

}

}

if (flag == 0)

{

cout<<target<<"Not found!";

}

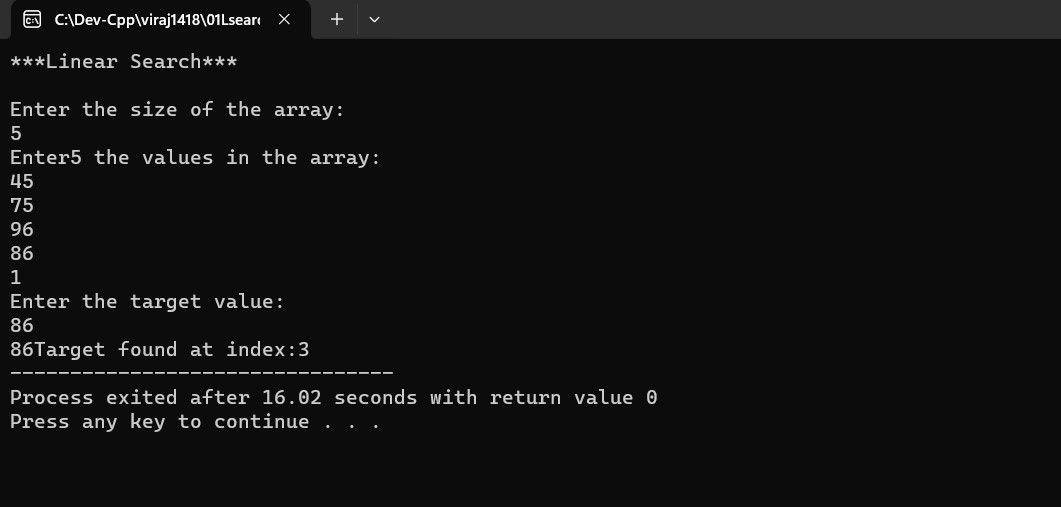
else

{

cout<<target<<"Target found at index:"<<i;

}

}//end of main

**Output:** 

|  |  |
| --- | --- |
| **Program No:** | **7** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 2: Searching – Binary Search** |
| **Objective :** | **Binary Searching** |

**Source Code:**

/\*Binary search

unit 5: tree\*/

#include<iostream>

#include<conio.h>

using namespace std;

/\*node template\*/

class BSTNode

{

public:

int data;

BSTNode \*right;

BSTNode \*left;

};

/\* list-based template\*/

class BST

{

BSTNode \*root;

int cnt;

public:

BST()

{

root = NULL;

cnt = 0;

}

void Insert(int x);

void Search(int x);

void Display();

void Inorder(BSTNode \*p);

void Preorder(BSTNode \*p);

void Postorder(BSTNode \*p);

void Count();

void FindMax();

void FindMin();

};

//functions

void BST::Insert(int x)

{

//make a new node t

BSTNode \*t = new BSTNode;

t->data = x;

t->right = NULL;

t->left = NULL;

//

if(root == NULL)

{

root = t;

cnt++;

return;

}

//Traverse

BSTNode \*tmp = root;

BSTNode \*prev = NULL;

while( tmp != NULL)

{

prev = tmp;

if(t->data < tmp->data)

{

tmp = tmp->left;

}

else if (t->data > tmp->data)

{

tmp = tmp->right;

}

else

{

cout<<" Duplicate values not aallowed";

getch();

}

}//end of while

//Attach t to prev

if(t->data < prev->data)

{

prev->left =t;

}

else

{

prev->right =t;

}

cnt++;

}//end of insert

void BST :: Display()

{

cout <<"Inorder traversal: ";

Inorder(root);

cout << endl;

cout <<"Preorder traversal: ";

Preorder(root);

cout << endl;

cout <<"Postorder traversal: ";

Postorder(root);

cout << endl;

}

void BST :: Inorder (BSTNode \* p)

{

if(p!= NULL)

{

Inorder (p->left);

cout<< p->data <<" ";

Inorder (p->right);

}

}

void BST :: Preorder (BSTNode \* p)

{

if(p!= NULL)

{

cout<< p->data <<" ";

Preorder (p->left);

Preorder (p->right);

}

}

void BST :: Postorder (BSTNode \* p)

{

if(p!= NULL)

{

Postorder (p->left);

Postorder (p->right);

cout<< p->data <<" ";

}

}

void BST :: Count()

{

//cnt is declared in the bst class

cout << " Number of nodes in BST is : "<<cnt;

}//end of count

void BST :: Search(int x)

{

BSTNode \*tmp = root;

int flag =0;

while( tmp != NULL)

{

if(x < tmp->data )

{

tmp= tmp->left;

}

else if (x > tmp->data )

{

tmp = tmp->right;

}

else //Match found

{

flag=1;

break;

}

}

if(flag ==0 )

{

cout <<"the number is not found : ";

}

else

{

cout << x << "number is found! ";

}

}

void BST :: FindMax()

{

BSTNode \*tmp = root;

while(tmp->right != NULL)

{

tmp = tmp->right;

}

cout << "Maximum value in the BST: "<<tmp->data;

}

void BST :: FindMin()

{

BSTNode \*tmp = root;

while(tmp->left!=NULL)

{

tmp = tmp->left;

}

cout << " Minimum value in the bst: " <<tmp->data;

}

//main menu

int main()

{

int ch, num;

BST b;

while(1)

{

system("cls");

cout<<"\*\*\* Binary Search Tree \*\*\*\n\n";

cout<<"1. Insert\n";

cout<<"2. Search\n";

cout<<"3. Display\n";

cout<<"4. Count\n";

cout<<"5. FindMax\n";

cout<<"6. FindMin\n";

cout<<"7. Exit\n";

cout<<"Enter an Options\n";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Insert:\n";

cin>>num;

b.Insert(num);

getch();

break;

case 2:

cout<<"Search";

cin>>num;

b.Search(num);

getch();

break;

case 3:

cout<<"Display\n";

b.Display();

getch();

break;

case 4:

cout<<"Count";

b.Count();

getch();

break;

case 5:

b.FindMax();

getch();

break;

case 6:

cout<<"FindMin";

b.FindMin();

getch();

break;

case 7:

exit(1);

default:

cout<<"Incorrect choice";

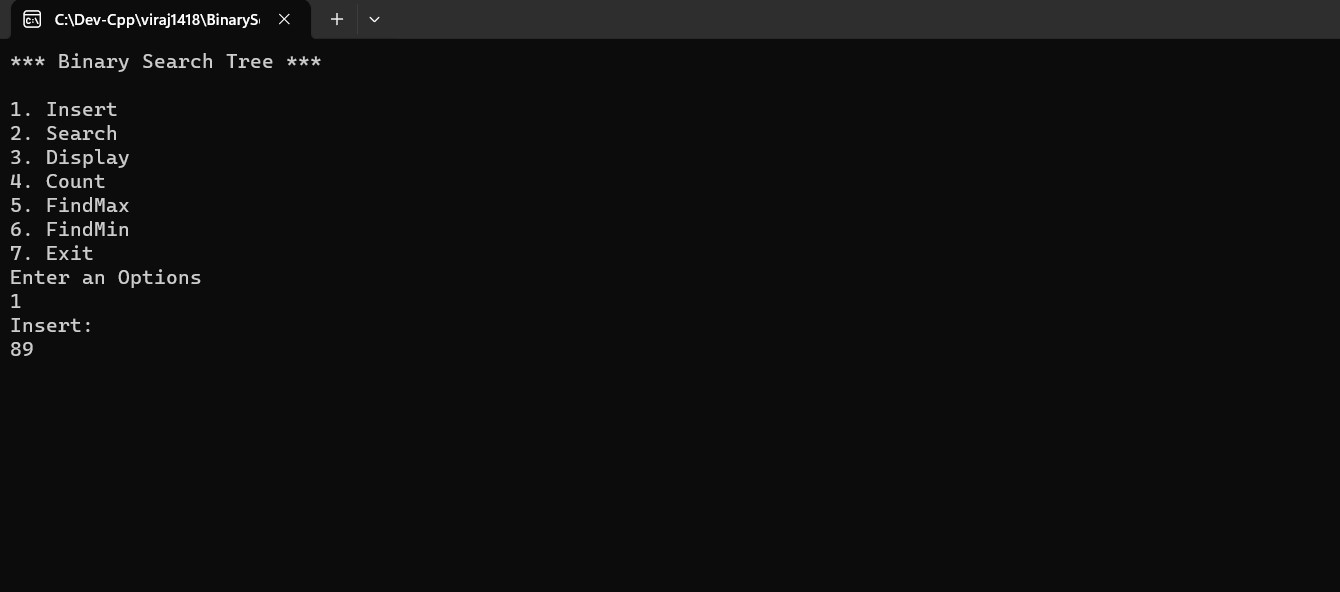
}

}

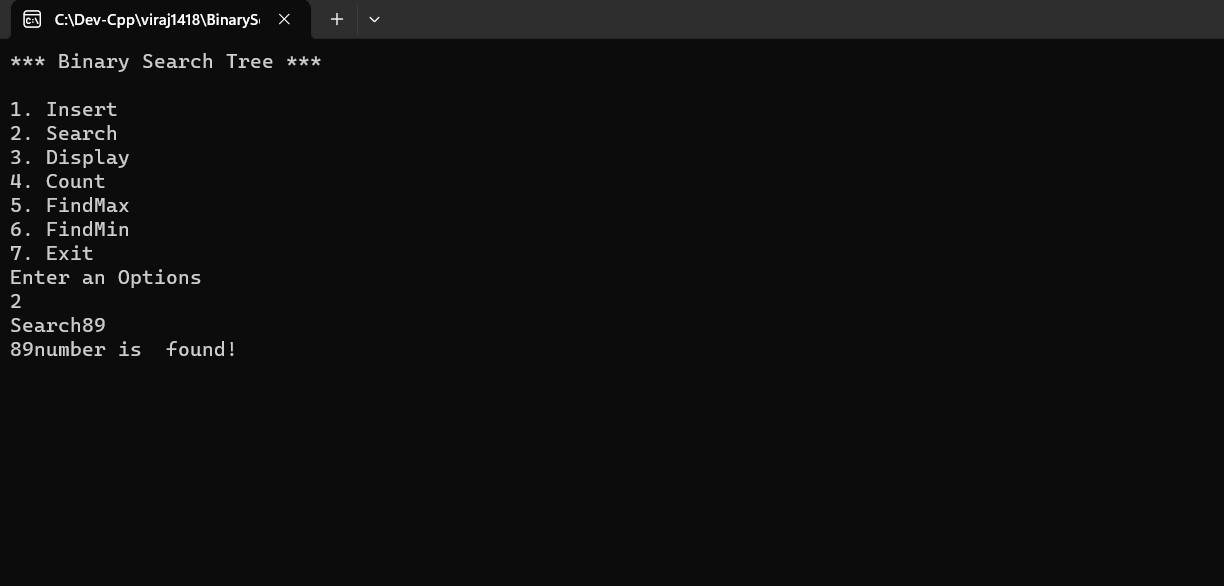
}//end of main

**Output:**

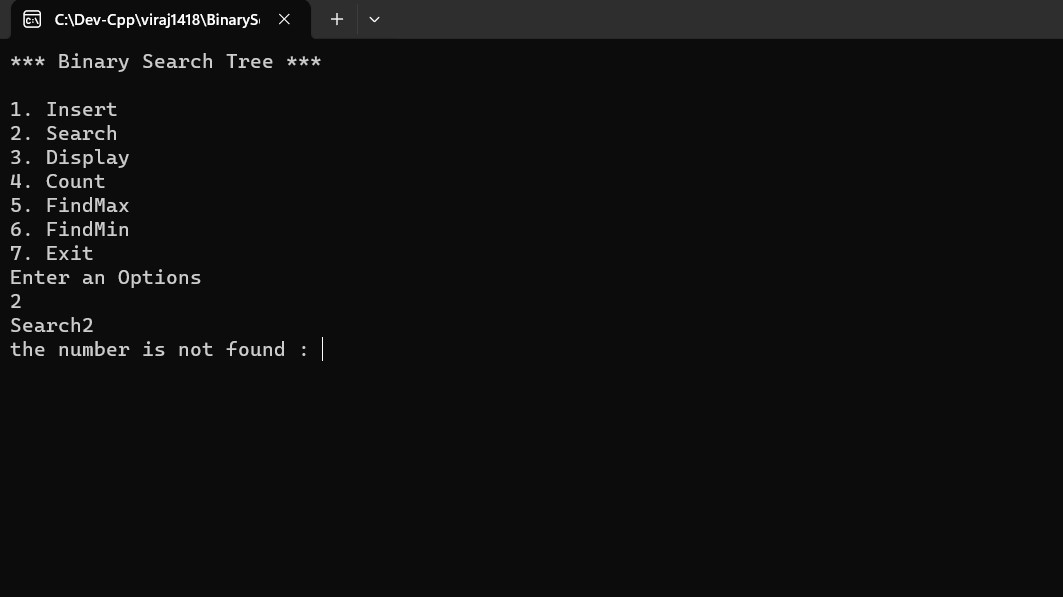
**Insert a node –**



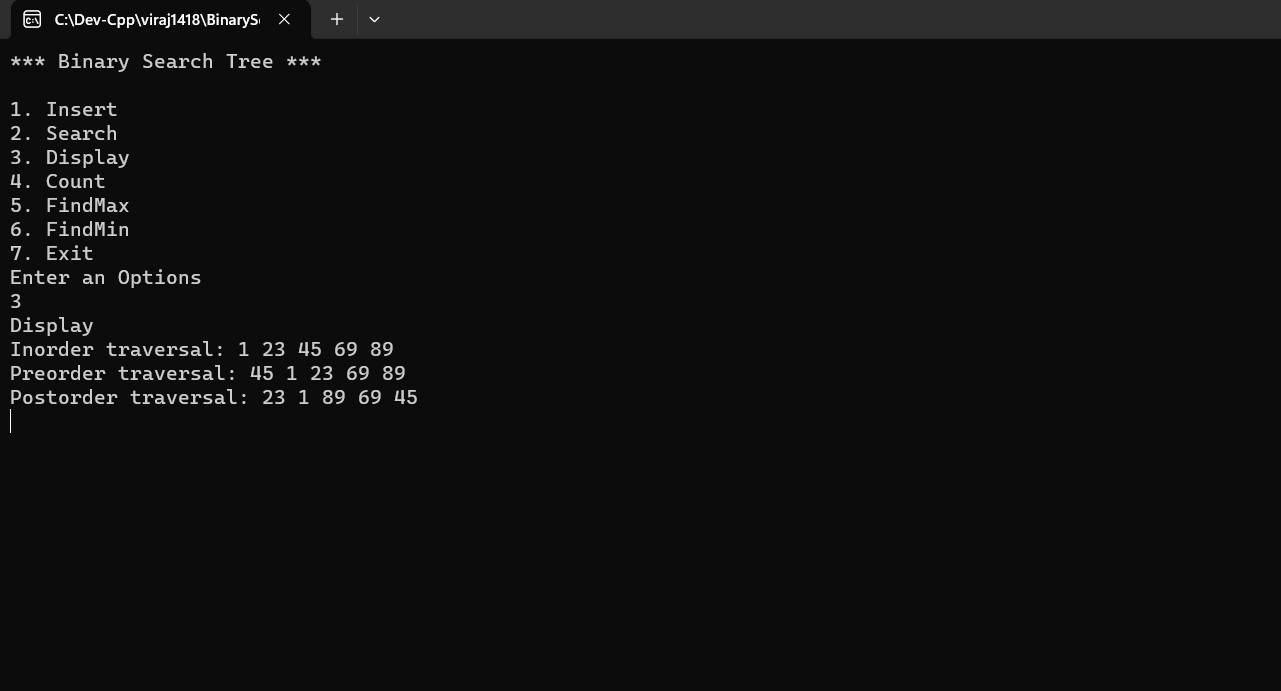
**Search a node -**



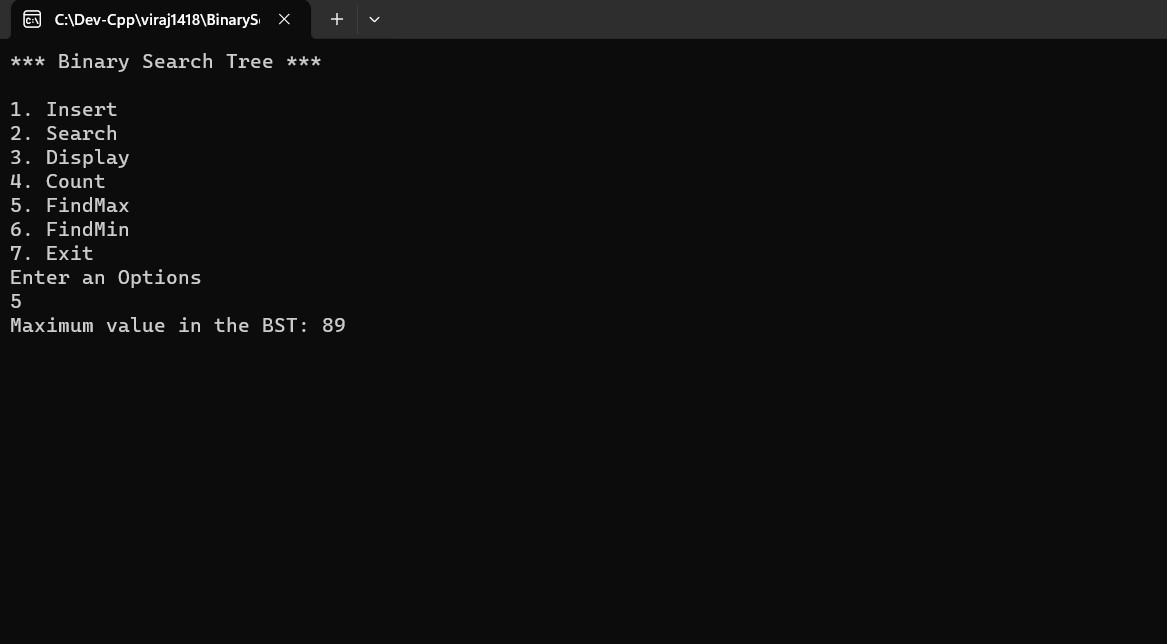
**If node is not found –**



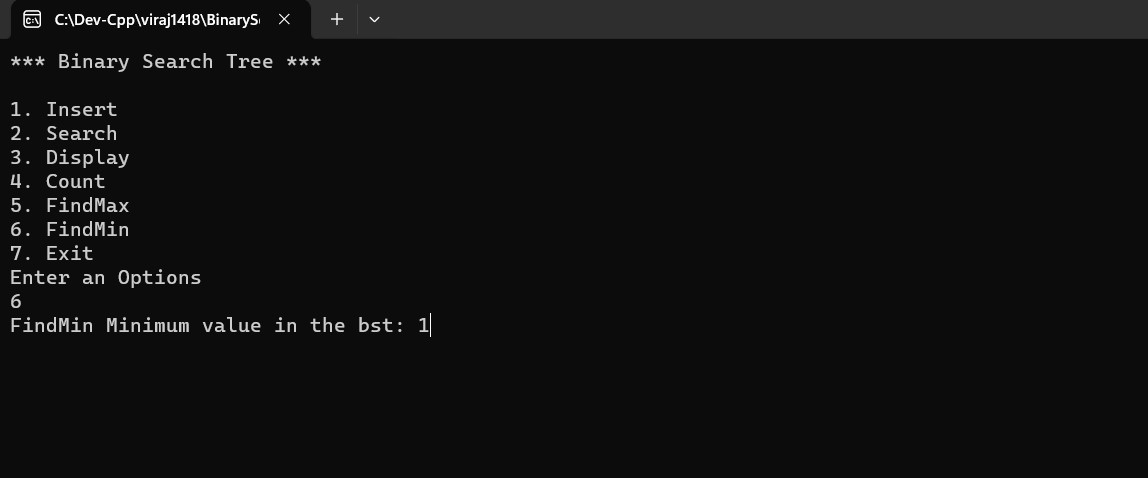
**Display the nodes –**



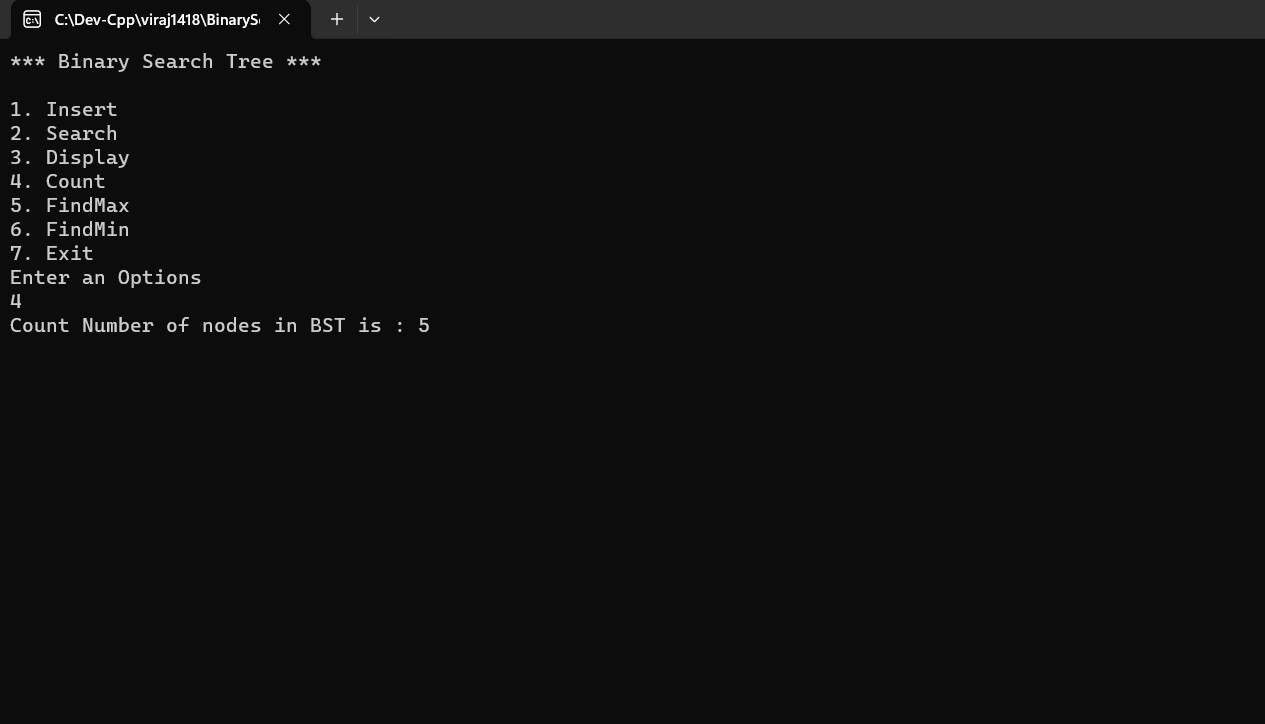
**Find the maximum node -**



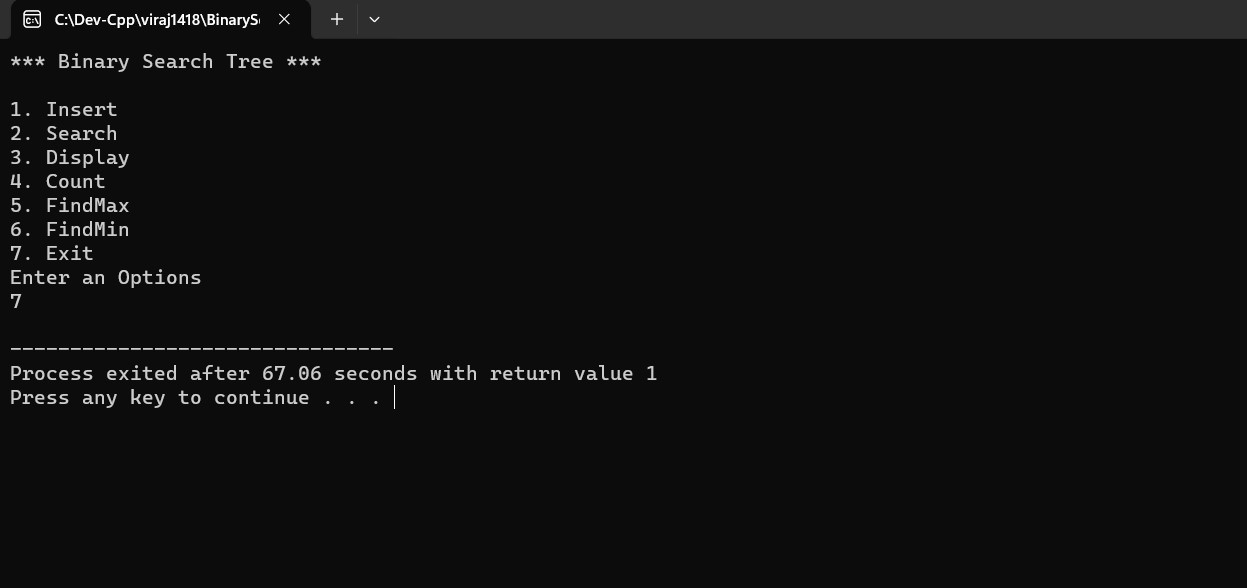
**Find the minimum node –**



**Count-**



**Exit –**



|  |  |
| --- | --- |
| **Program No:** | **8** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 2: Hashing – Modulo Division with Linear Probe** |
| **Objective :** | **Modulo division with linear probe** |

**Source Code:**

/\*

Name :

Roll NO:

Unit 2:Hashing

Program: Modulo Division Hashing with linear Probe for collisions

\*/

#include<iostream>

#include<conio.h>

#define MAX 10

#define EMPTY -1

using namespace std;

//Node Template

//Array Based

class Hash

{

int h[MAX];

public:

Hash()

{

int i;

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

{

h[i]= EMPTY;

}

}

void Insert(int key);

void Display();

void Search(int skey);

};

//3.Functions

void Hash :: Insert(int key)

{

//HASH FNC

int addr =key% MAX; //Module Division Hash function

int oaddr = addr;

if(h[addr] == EMPTY) //addr is available

{

h[addr] = key;

cout<< key << " is inserted at address " << addr << endl;

}

else

{

//collision!!

do

{

addr =(addr+1) % MAX;

if(addr == oaddr)

{

cout<< "Hash Table is Full!";

return;

}

}

while(h[addr]!=EMPTY);

h[addr] = key;

cout<< key << " is inserted at address " << addr << endl;

}//end of if-else

}

void Hash :: Display()

{

int i ;

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

{

cout << "h[ " << i << "] = ";

if(h[i] == EMPTY )

{

cout << "EMPTY " << endl;

}

else

{

cout << h[i] <<endl;

}

}

}

void Hash :: Search(int skey)

{

int saddr = skey%MAX;

int oaddr = saddr;

if(h[saddr]==skey)

{

cout << "Key " << skey << " is found at address" << saddr << endl;

}

else

{

do

{

saddr = (saddr+1) %MAX;

if(saddr == oaddr)

{

break;

}

}

while(h[saddr]!=skey);

if(h[saddr] == skey)

{

cout <<"key "<< skey <<"is found at address"<< saddr << endl;

}

else

{

cout <<"Key "<< skey << "is not found" << endl;

}

}

}

//4.Menu

int main()

{

int ch,num;

Hash h;

while(1)

{

system("cls");

cout<< "\*\*\* Hashing with Modulo Division & Linear Probe \*\*\*\n\n";

cout<< "1.Insert a value in the hashtable\n";

cout<< "2.Display the hashtable\n";

cout<< "3.Search for the key in the hashtable\n";

cout<< "4.Exit\n\n";

cout<<"Enter your choice : ";

cin>>ch;

switch(ch)

{

case 1:

cout<< "Enter a key: ";

cin >> num;

h.Insert(num);

getch();

break;

case 2:

h.Display();

getch();

break;

case 3:

cout<< "Enter a key: ";

cin >> num;

h.Search(num);

getch();

break;

case 4:

exit(1);

default:

cout <<" Incorrect choice! ";

getch();

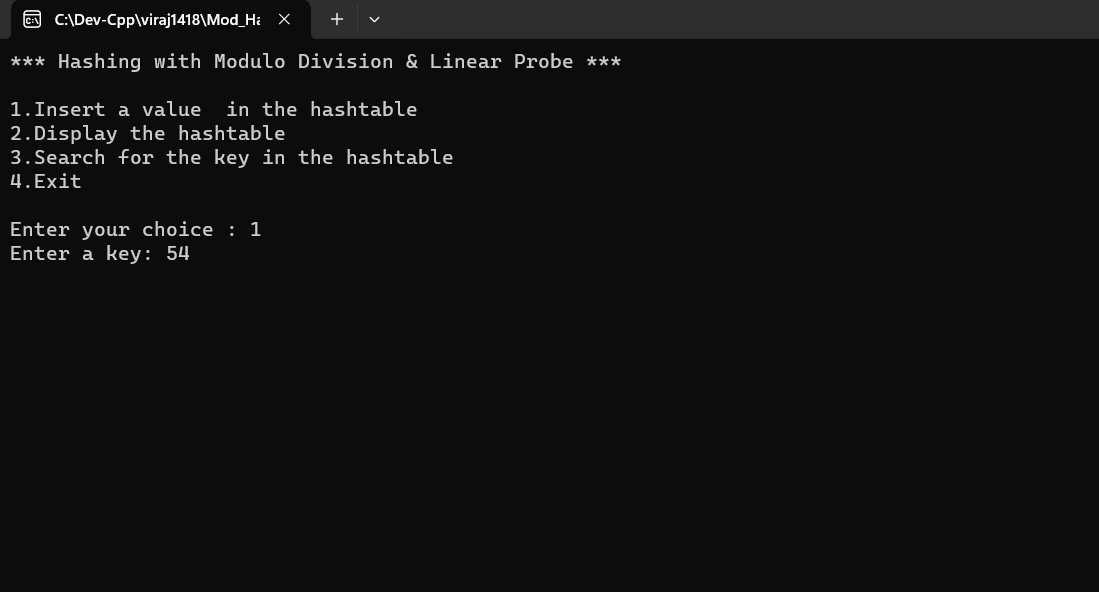
}//end of switch

}

}

**Output:**

**Inserting an element into the hash table –**



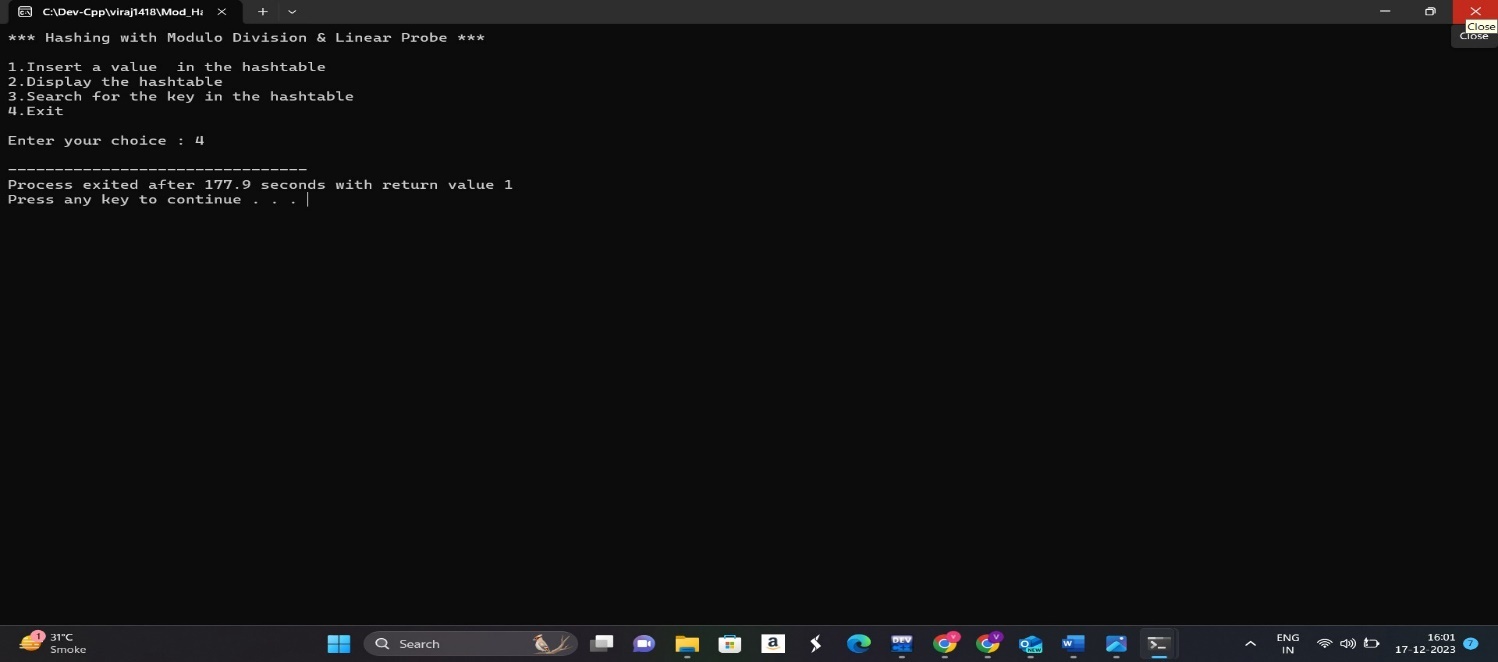
**Display the hash table –**



**Searching for an element in the hash table –**



Exit-



|  |  |
| --- | --- |
| **Program No:** | **9** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 2: Hashing – Digit Extraction with linear probe** |
| **Objective :** | **Modulo division with linear probe** |

Source code:

#include <iostream>

#include <conio.h>

#define MAX 10

#define EMPTY -1

using namespace std;

// Node Template - NOT REQUIRED

// Array Based Template for Hash Table

class Hash

{

int h[MAX];

public:

Hash()

{

int i;

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

{

h[i] = EMPTY;

}

}

void Insert(int key);

void Display();

void Search(int skey);

};

// Functions

void Hash ::Insert(int key)

{

// Hash Function

int digit1 = (key / 1000) % 10; // Extract 1st digit - Thousands place

int digit3 = (key / 10) % 10; // Extract 3rd digit - Tens place

int addr = (digit1 \* 10 + digit3) % MAX; // Digit Extraction Hash Function

int oaddr = addr;

if (h[addr] == EMPTY) // addr is available

{

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

else

{

// Collision!!

do

{

addr = (addr + 1) % MAX;

if (addr == oaddr)

{

cout << "Hash Table is Full!";

return;

}

} while (h[addr] != EMPTY);

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

}

void Hash ::Display()

{

int i;

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

{

cout << "h[" << i << "] = ";

if (h[i] == EMPTY)

{

cout << "EMPTY" << endl;

}

else

{

cout << h[i] << endl;

}

}

}

void Hash ::Search(int skey)

{

int saddr = skey % MAX;

int oaddr = saddr;

if (h[saddr] == skey)

{

cout << "Key " << skey << " is found at address " << saddr << endl;

}

else

{

do

{

saddr = (saddr + 1) % MAX;

if (saddr == oaddr)

{

break;

}

} while (h[saddr] != skey);

if (h[saddr] == skey)

{

cout << "Key " << skey << " is found at address " << saddr <<

endl;

}

else

{

cout << "Key " << skey << " is not found!" << endl;

}

}

}

// Menu

int main()

{

int ch, num;

Hash h;

while (1)

{

system("cls");

cout << "\*\*\* Hashing with Digit Extraction & Linear Probe \*\*\*\n\n";

cout << "1. Insert a value in the Hash Table\n";

cout << "2. Display the Hash Table\n";

cout << "3. Search for a key in the Hash Table\n";

cout << "4. Exit\n\n";

cout << "Enter your choice: ";

cin >> ch;

switch (ch)

{

case 1:

cout << "Enter key: ";

cin >> num;

h.Insert(num);

getch();

break;

case 2:

h.Display();

getch();

break;

case 3:

cout << "Enter key to Search: ";

cin >> num;

h.Search(num);

getch();

break;

case 4:

exit(1);

default:

cout << "Incorrect Choice!";

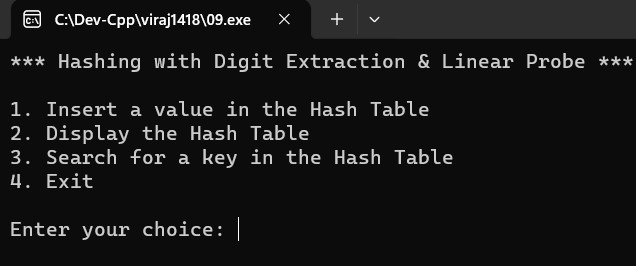
getch();

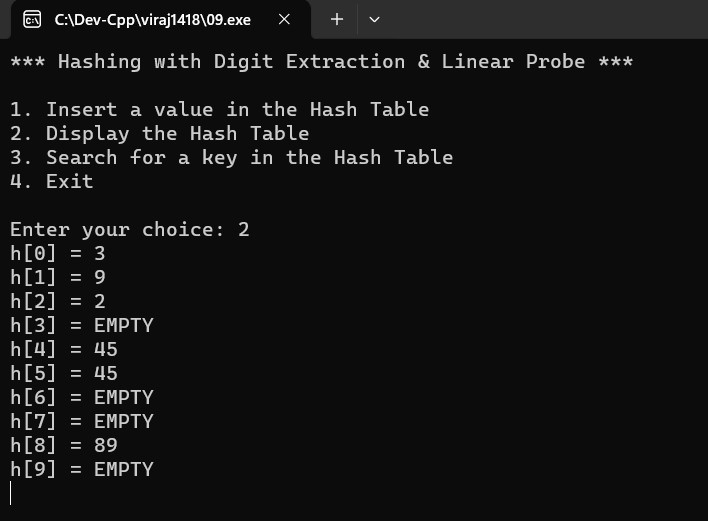
} // end of switch

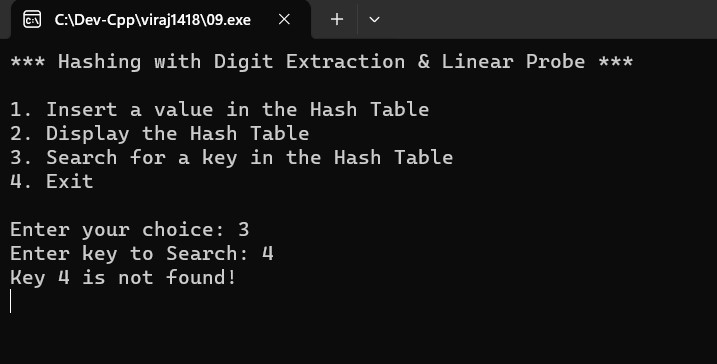
} // end of while

} // end of main

Output:







|  |  |
| --- | --- |
| **Program No:** | 10 |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 2: Hashing – Fold shift with linear probe** |
| **Objective :** | **Modulo division with linear probe** |

Source code:

#include <iostream>

#include <conio.h>

#define MAX 10

#define EMPTY -1

using namespace std;

// Node Template - NOT REQUIRED

// Array Based Template for Hash Table

class Hash

{

int h[MAX];

public:

Hash()

{

int i;

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

{

h[i] = EMPTY;

}

}

void Insert(int key);

void Display();

void Search(int skey);

};

// Functions

void Hash ::Insert(int key)

{

// Hash Function

int addr = (((key / 10000) % 100) + ((key / 100) % 100) + (key % 100)) % 100;

int oaddr = addr;

if (h[addr] == EMPTY) // addr is available

{

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

else

{

// Collision!!

do

{

addr = (addr + 1) % MAX;

if (addr == oaddr)

{

cout << "Hash Table is Full!";

return;

}

}

while (h[addr] != EMPTY);

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

}

void Hash ::Display()

{

int i;

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

{

cout << "h[" << i << "] = ";

if (h[i] == EMPTY)

{

cout << "EMPTY" << endl;

}

else

{

cout << h[i] << endl;

}

}

}

void Hash ::Search(int skey)

{

int saddr = skey % MAX;

int oaddr = saddr;

if (h[saddr] == skey)

{

cout << "Key " << skey << " is found at address " << saddr << endl;

}

else

{

do

{

saddr = (saddr + 1) % MAX;

if (saddr == oaddr)

{

break;

}

}

while (h[saddr] != skey);

if (h[saddr] == skey)

{

cout << "Key " << skey << " is found at address " << saddr <<

endl;

}

else

{

cout << "Key " << skey << " is not found!" << endl;

}

}

}

// Menu

int main()

{

int ch, num;

Hash h;

while (1)

{

system("cls");

cout << "\*\*\* Hashing with Fold Shift & Linear Probe \*\*\*\n\n";

cout << "1. Insert a value in the Hash Table\n";

cout << "2. Display the Hash Table\n";

cout << "3. Search for a key in the Hash Table\n";

cout << "4. Exit\n\n";

cout << "Enter your choice: ";

cin >> ch;

switch (ch)

{

case 1:

cout << "Enter key: ";

cin >> num;

h.Insert(num);

getch();

break;

case 2:

h.Display();

getch();

break;

case 3:

cout << "Enter key to Search: ";

cin >> num;

h.Search(num);

getch();

break;

case 4:

exit(1);

default:

cout << "Incorrect Choice!";

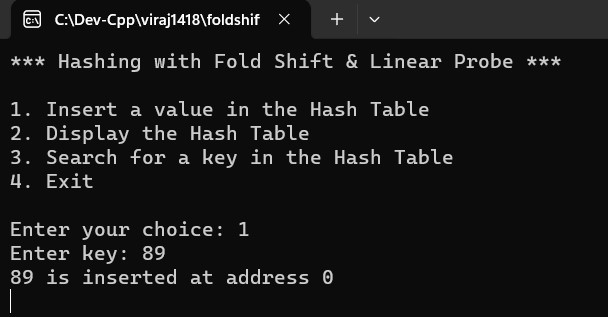
getch();

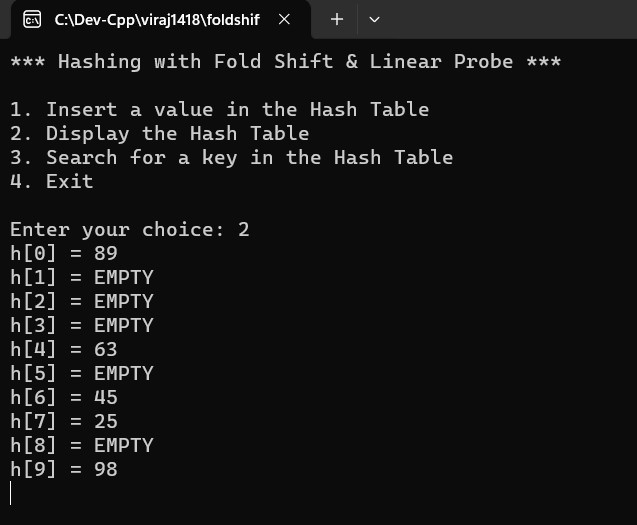
} // end of switch

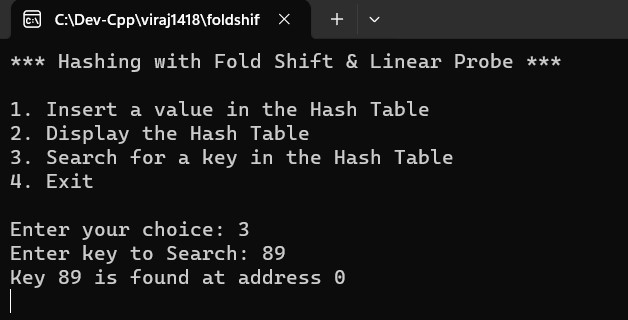
} // end of while

} // end of main

Output:







|  |  |
| --- | --- |
| **Program No:** | 11 |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 2: Hashing – Fold boundry with linear probe** |
| **Objective :** | **Modulo division with linear probe** |

Source code:

#include <iostream>

#include <conio.h>

#define MAX 10

#define EMPTY -1

using namespace std;

// Node Template - NOT REQUIRED

// Array Based Template for Hash Table

class Hash

{

int h[MAX];

public:

Hash()

{

int i;

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

{

h[i] = EMPTY;

}

}

void Insert(int key);

void Display();

void Search(int skey);

};

// Functions

// reverse function

int revNum(int num)

{

int rev = 0;

if (num < 10)

{

rev = num \* 10;

return rev;

}

while (num != 0)

{

int digit = num % 10;

rev = rev \* 10 + digit;

num /= 10;

}

return rev;

}

void Hash ::Insert(int key)

{

// Hash Function

int addr = (revNum((key / 10000) % 100) + ((key / 100) % 100) + revNum(key %

100)) % 100;

int oaddr = addr;

if (h[addr] == EMPTY) // addr is available

{

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

else

{

// Collision!!

do

{

addr = (addr + 1) % MAX;

if (addr == oaddr)

{

cout << "Hash Table is Full!";

return;

}

}

while (h[addr] != EMPTY);

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

}

void Hash ::Display()

{

int i;

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

{

cout << "h[" << i << "] = ";

if (h[i] == EMPTY)

{

cout << "EMPTY" << endl;

}

else

{

cout << h[i] << endl;

}

}

}

void Hash ::Search(int skey)

{

int saddr = skey % MAX;

int oaddr = saddr;

if (h[saddr] == skey)

{

cout << "Key " << skey << " is found at address " << saddr << endl;

}

else

{

do

{

saddr = (saddr + 1) % MAX;

if (saddr == oaddr)

{

break;

}

}

while (h[saddr] != skey);

if (h[saddr] == skey)

{

cout << "Key " << skey << " is found at address " << saddr <<

endl;

}

else

{

cout << "Key " << skey << " is not found!" << endl;

}

}

}

// Menu

int main()

{

int ch, num;

Hash h;

while (1)

{

system("cls");

cout << "\*\*\* Hashing with Fold Boundary & Linear Probe \*\*\*\n\n";

cout << "1. Insert a value in the Hash Table\n";

cout << "2. Display the Hash Table\n";

cout << "3. Search for a key in the Hash Table\n";

cout << "4. Exit\n\n";

cout << "Enter your choice: ";

cin >> ch;

switch (ch)

{

case 1:

cout << "Enter key: ";

cin >> num;

h.Insert(num);

getch();

break;

case 2:

h.Display();

getch();

break;

case 3:

cout << "Enter key to Search: ";

cin >> num;

h.Search(num);

getch();

break;

case 4:

exit(1);

default:

cout << "Incorrect Choice!";

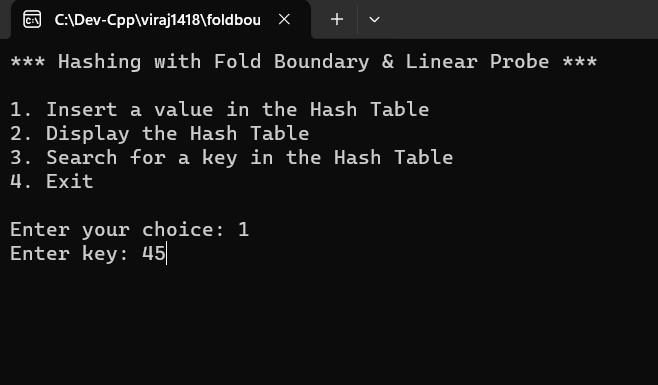
getch();

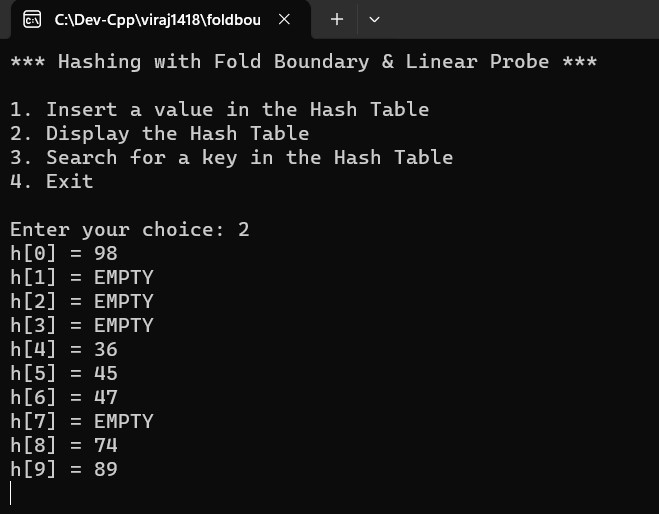
} // end of switch

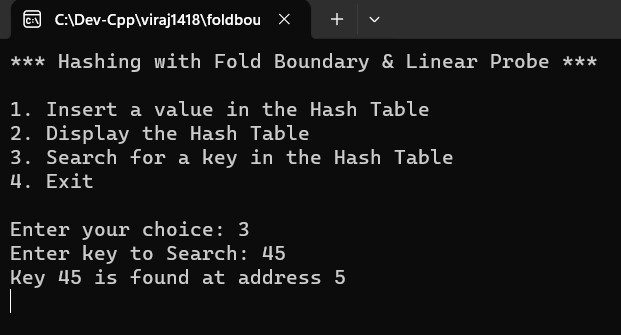
} // end of while

} // end of main

Output:







|  |  |
| --- | --- |
| **Program No:** | **12** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 3: Stacks – Array Based Stacks** |
| **Objective :** | **Array based stacks** |

**Source Code:**

/\* Name : Viraj Ghogale Roll No : 1418 unit 3 : Stacks

Program : Array Based Stack \*/

#include<iostream>

#include<conio.h>

#define SIZE 5

using namespace std;

/\* 1. Node Template - NOT REQUIRED as it is array based \*/

/\* 2. Array Based Stack \*/ class AStack { int A[SIZE]; int tos;

public:

AStack()

{

tos = -1;

}

void Push(int x);

void Pop();

void Peek();

void Display();

int Full();

int Empty();

};

/\* 3. Functions \*/

int AStack :: Full()

{

if(tos == SIZE-1) {

return 1;

}

else

{

return 0;

}

}

int AStack :: Empty()

{

if(tos == -1)

{

return 1;

}

else

{

return 0;

}

}

void AStack :: Push(int x) {

if(Full())

{

cout << "Stack OverFlow!"; return;

}

tos++;

A[tos] = x;

}

void AStack :: Pop()

{ if(Empty()) { cout << "Stack Underflow";

return; }

int tmp = A[tos];

tos--;

cout << "Element popped is " << tmp;

}

void AStack :: Peek()

{ if(Empty()) { cout << "Underflow"; return; }

cout << "Element at the tos is: " << A[tos];

}

void AStack :: Display()

{ if(Empty()) { cout << "Stack underflow";

return; }

int i;

cout << "Stack contains: \n"; for(i=tos; i>=0; i--) { cout << A[i] << endl;

}

}

/\* Menu \*/ int main() { AStack s; int num, ch;

while(1) {

system("cls");

cout << "\*\*\*\*\* ARRAY BASED STACK \*\*\*\*\*\n\n";

cout << "1. Push an Element on the stack\n\n";

cout << "2. Pop an Element from the stack\n\n";

cout << "3. Peek at the stack\n\n";

cout << "4. Display the Stack\n\n";

cout << "5. Exit\n\n";

cout << "Enter the choice\n";

cin >> ch;

switch(ch)

{

case 1 :

cout << "Enter a Element\n";

cin >> num;

s.Push(num);

getch();

break;

case 2 :

s.Pop();

getch();

break;

case 3 :

s.Peek();

getch();

break;

case 4 :

s.Display();

getch();

break;

case 5 :

exit(0);

default : cout << "Incorrect Choice!";

getch();

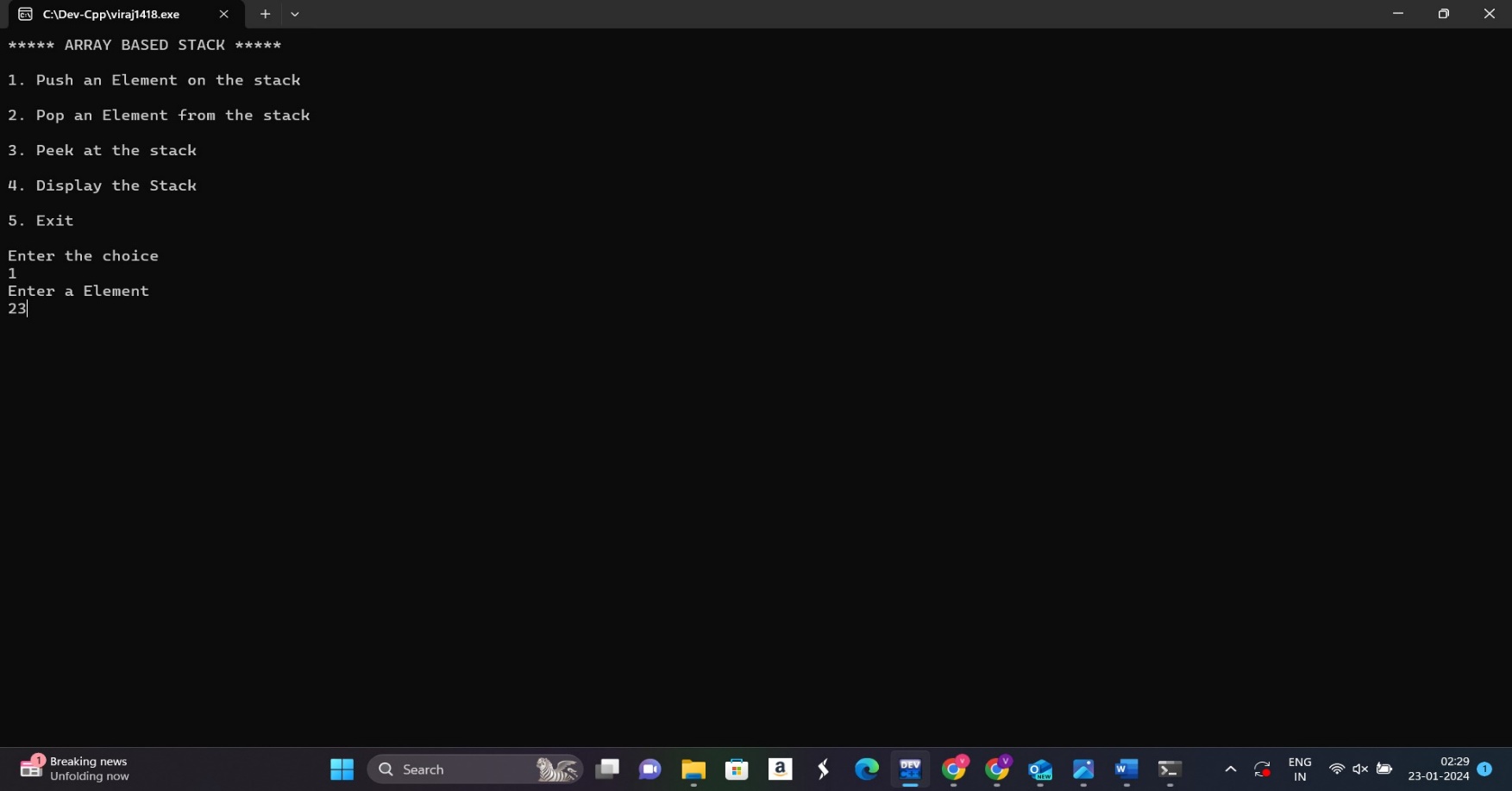
}

}

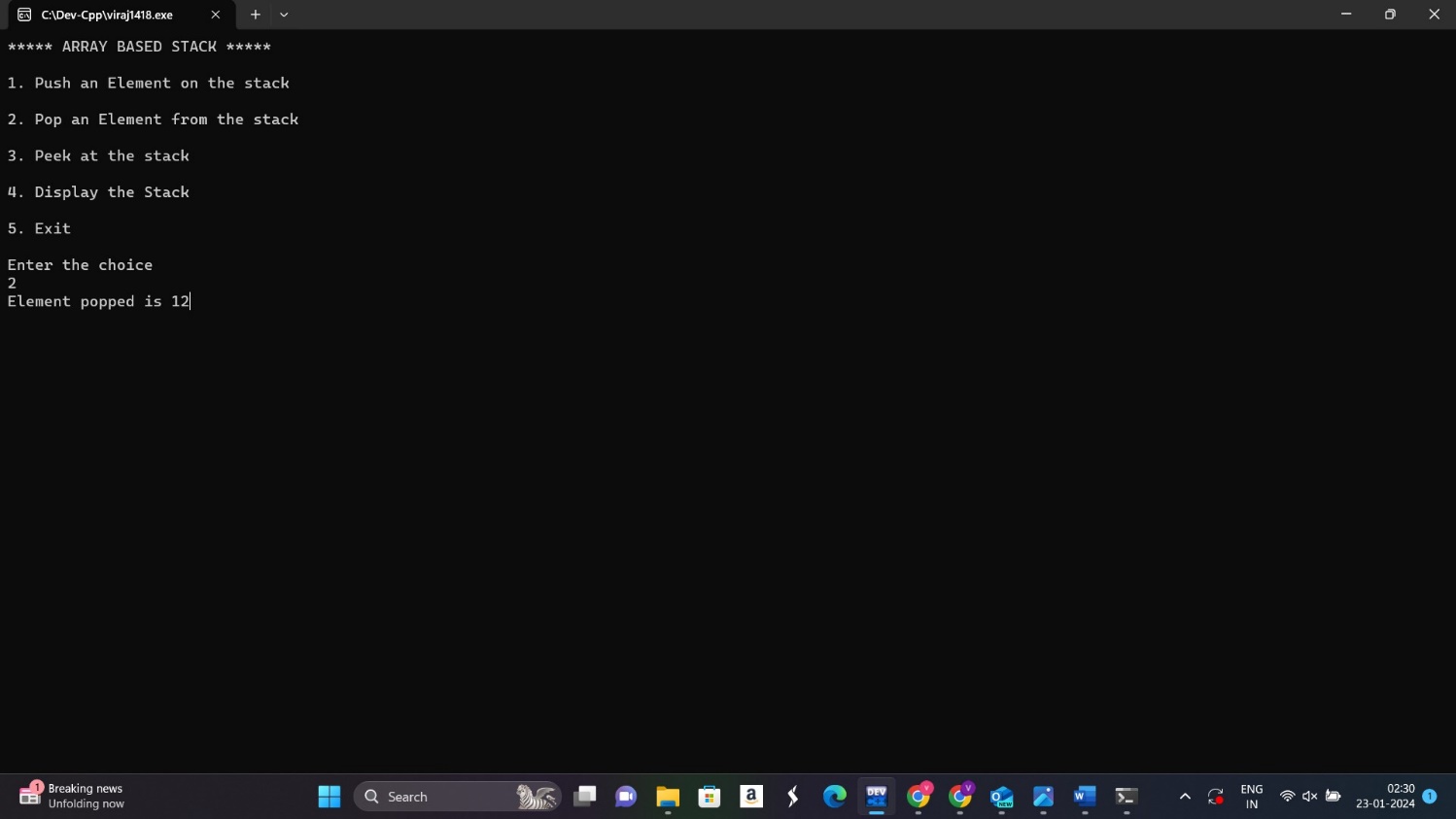
}

**Output:**

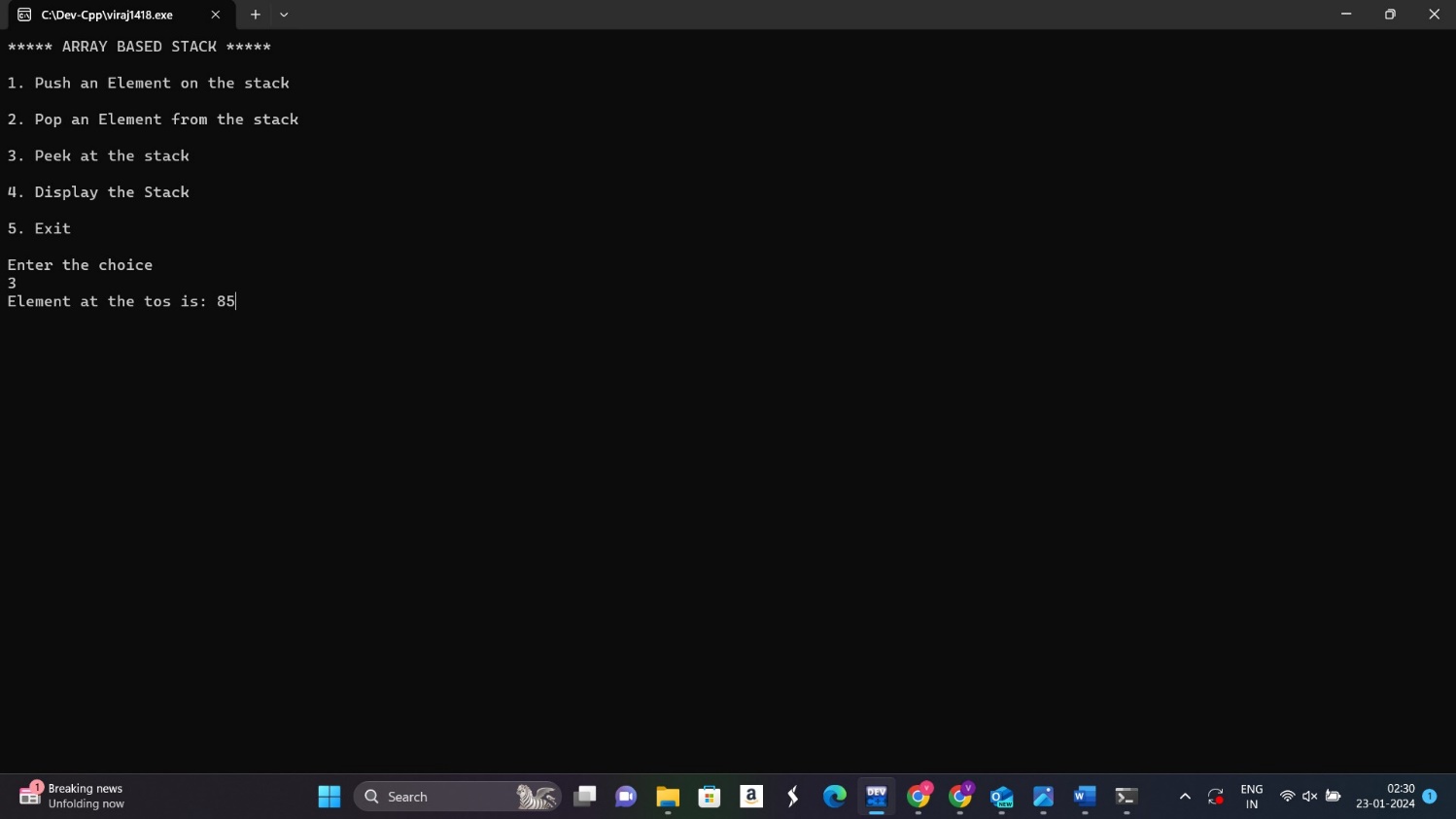
**Inserting an element –**



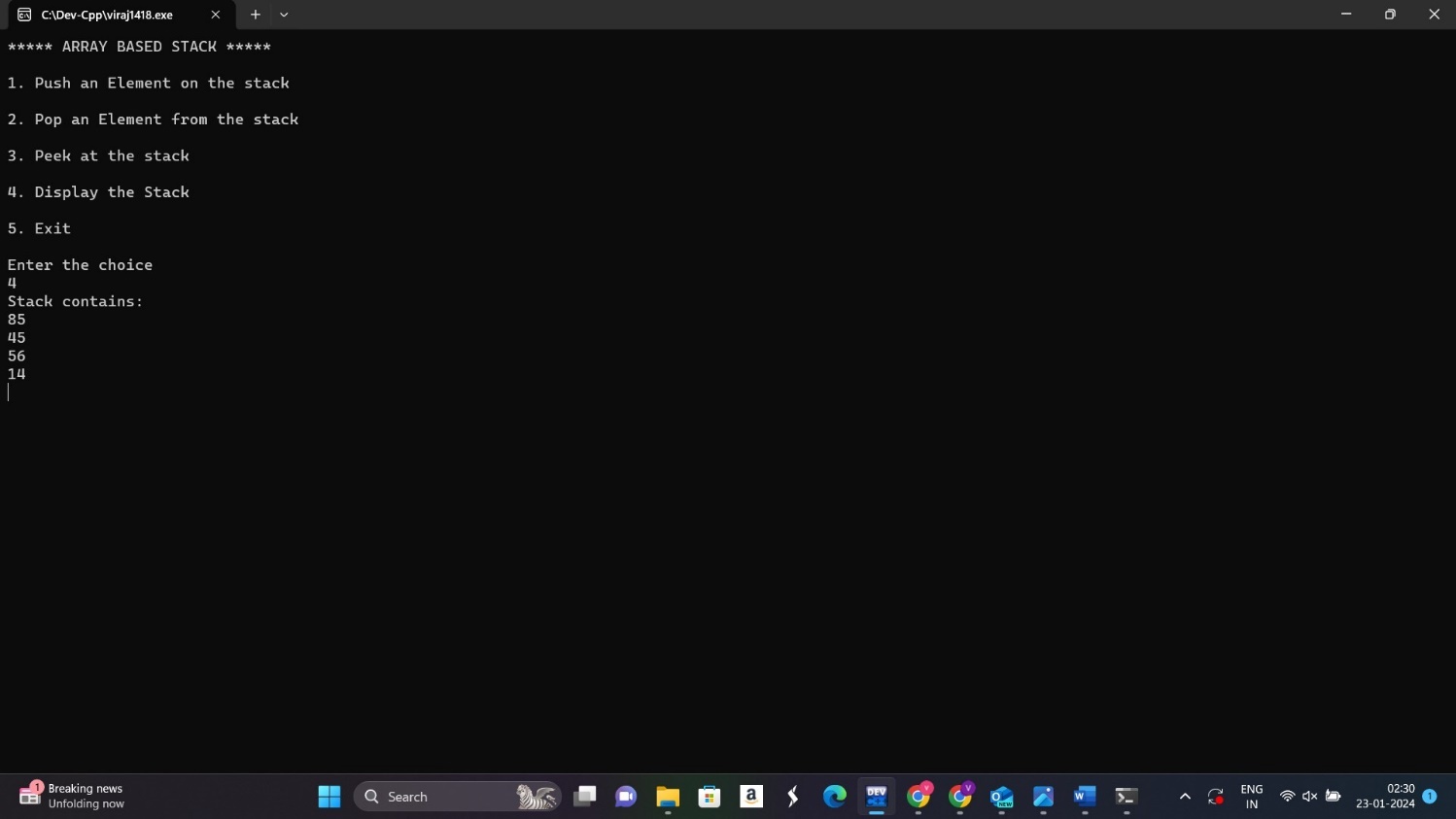
**To pop an element –**



**To peek the stack –**



**Display the stack –**



|  |  |
| --- | --- |
| **Program No:** | **13** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 3: Stacks – List Based Stacks** |
| **Objective :** | **List based stacks** |

**Source Code:**

**/**\* Name : Viraj Ghogale

Roll No :1418

unit 3 : Stack

Program : List Based Stack \*/

#include<iostream>

#include<conio.h>

#define SIZE 5

using namespace std;

//1. Node template class SNode { public: int data; SNode \*next;

};

//2. List template class LStack {

SNode \*tos; public: LStack()

{

tos = NULL;

}

void Push(intx);

void Pop(int x); void Peek(int x);

void Display(int x); int Empty();

};

//3. Functions

void LStack :: Push(int x)

{

SNode \*t = new SNode;

t->data = x;

t->next = NULL;

if(tos == NULL) {

tos = t;

}

else {

t->next =tos;

tos = t;

}

} // END OF PUSH

void LStack :: Pop(int x)

{

if(tos==NULL)

{

cout << "Stack underflow!";

return;

}

SNode \*tmp = tos;

tos = tos->next;

cout << "Element popped is: " << tmp->data;

delete tmp;

} // END OF POP

void LStack :: Peek(int x)

{

if(tos==NULL)

{

cout << "Underflow"; return;

}

cout << "Element at the tos is: " << tos->data;

} // END OF PEEK

void LStack :: Display(int x)

{

if(tos==NULL)

{

cout << "Stackunderflow";

return;

}

SNode \*tmp = tos;

cout << "Stack contains: \n"; while(tmp!=NULL)

{

cout << tmp->data << endl;

tmp = tmp->next;

}

}

//4. Menu int main()

{

int num, ch;

LStack s;

while(1)

{ system("cls");

cout << "\*\*\*\*\* LIST BASED STACK \*\*\*\*\*\n\n";

cout << "1. Push an Element on the stack\n\n";

cout << "2. Pop an Element from the stack\n\n";

cout << "3. Peek at the stack\n\n";

cout << "4. Display the Stack\n\n";

cout << "5. Exit\n\n";

cout << "Enter the choice\n"; cin >> ch;

switch(ch) { case 1: cout << "Enter an element: "; cin >> num;

s.Push(num); getch();

break;

case 2:

s.Pop(num); getch(); break;

case 3:

s.Peek(num); getch();

break;

case 4:

s.Display(num); getch(); break;

case 5: exit(1);

default:

cout << "Incorrect choice !";

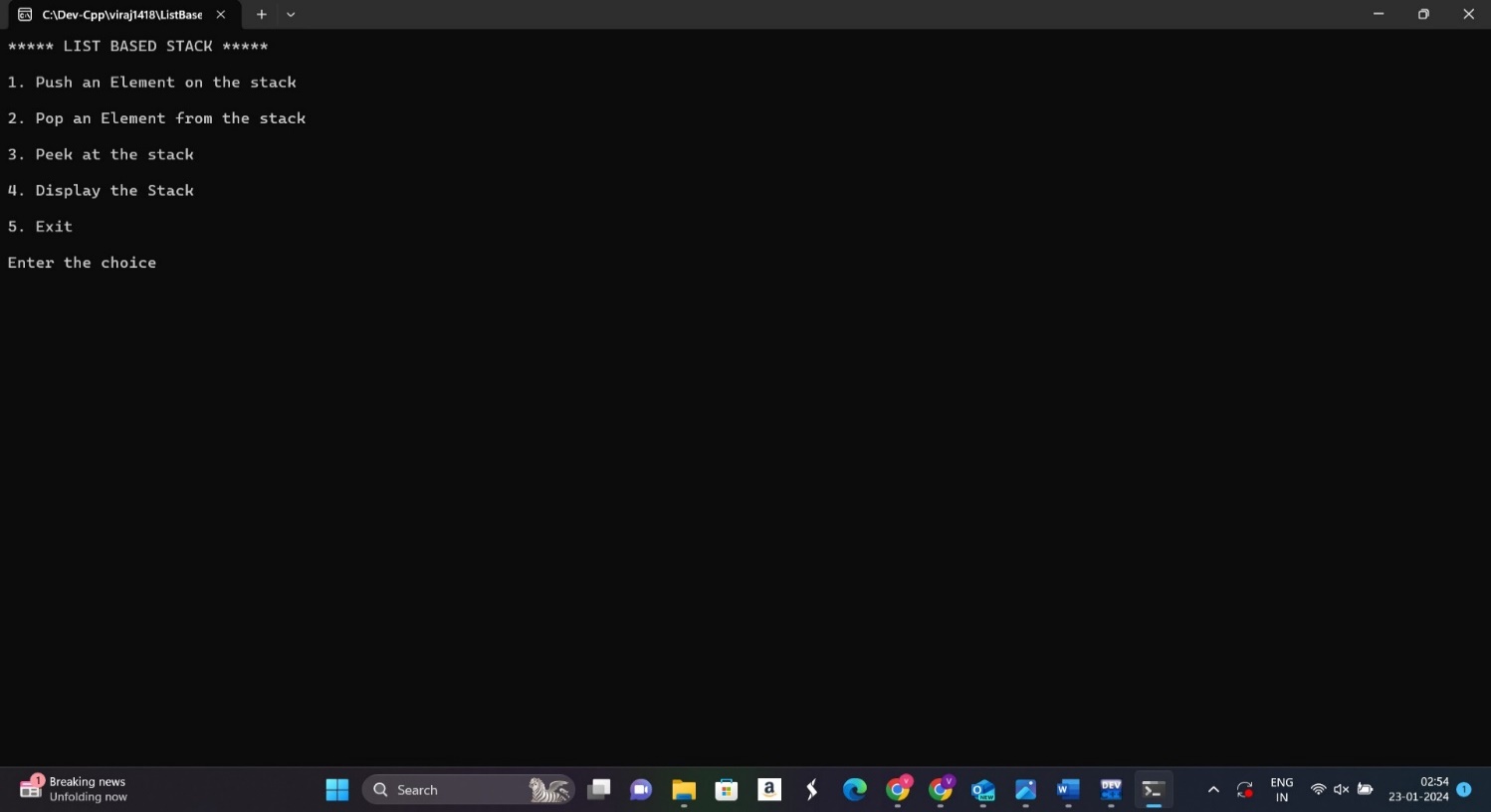
getch();

}

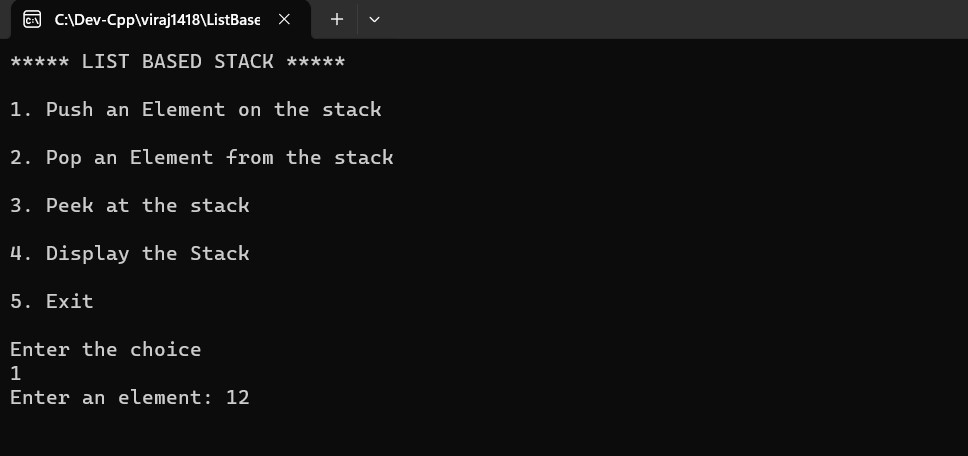
}

}

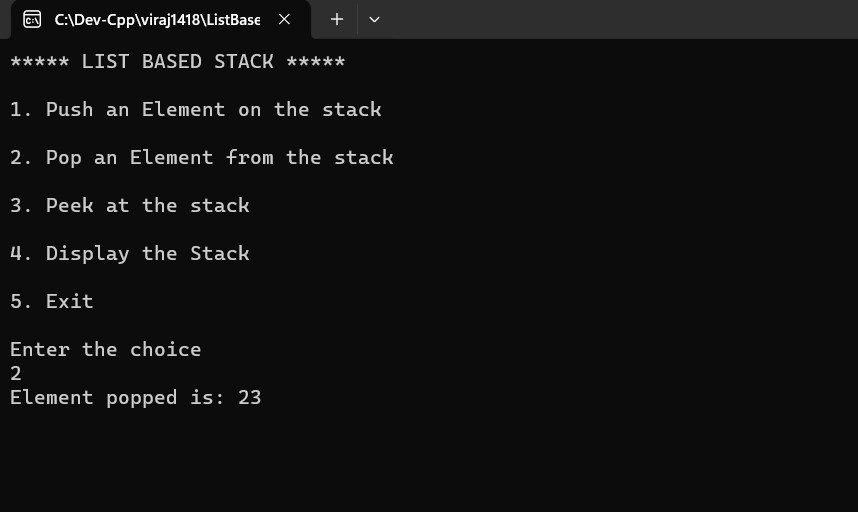
**Output:**



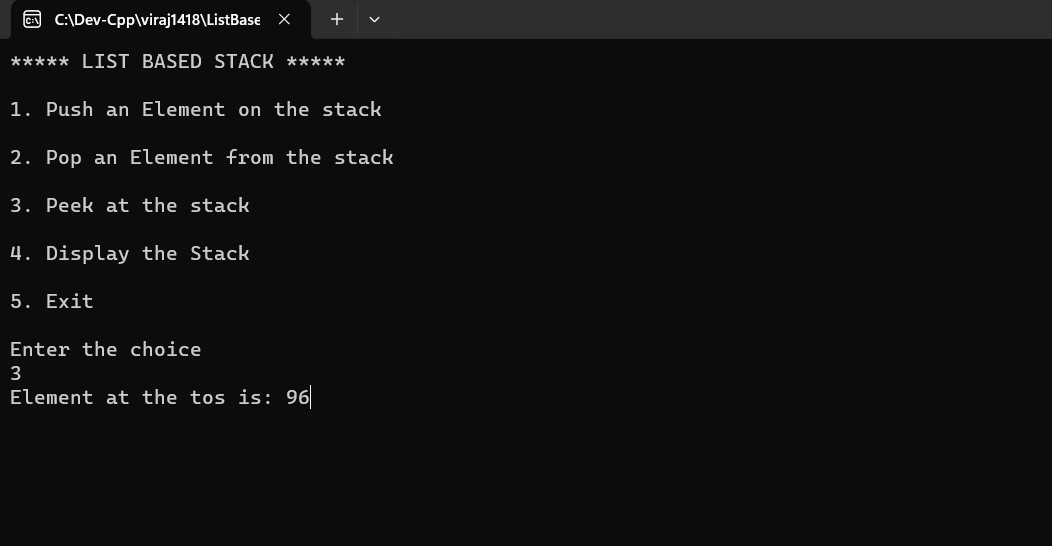
**To push an element into the stack –**



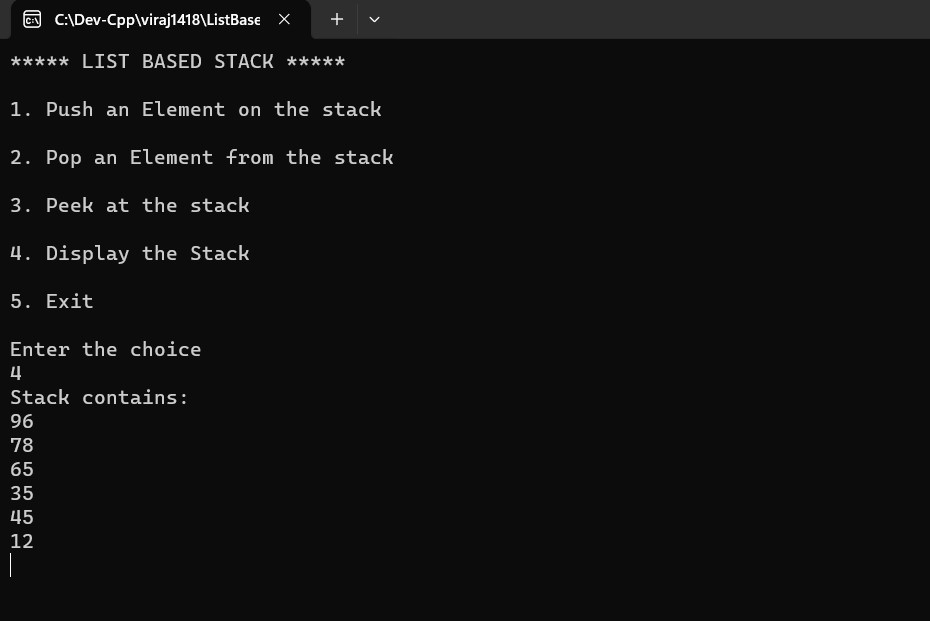
**pop an element –**



**To peek element at the stack –**



**To display the stack –**



|  |  |
| --- | --- |
| **Program No:** | **14** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 3: Stacks – Evaluation of Postfix Expression using Stacks** |
| **Objective :** | **Evaluation of Postfix Expression using Stacks** |
|  |  |

**Source code:**

**/\***

**Name:Viraj Ghogale**

**RollNo:1418**

**Program:Evaluation of postfix expression using stacks**

**\*/**

**Source Code:**

#include<iostream>

#include<conio.h>

#include<ctype.h>

#define SIZE 20

using namespace std;

class Stack

{

float A[SIZE];

int tos;

public: Stack() { tos = -1; }

void Push(float x);

float Pop();

int Full();

int Empty();

};

void Stack :: Push(float x)

{

if(Full())

{

cout << "Stack overflow";

return;

}

tos++;

A[tos] = x;

}

float Stack :: Pop()

{

if(Empty())

{

cout << "Stack underflow";

return -1;

}

float tmp = A[tos];

tos--;

return tmp;

}

int Stack :: Full()

{

return(tos==SIZE-1?:0);

}

int Stack :: Empty()

{ return(tos==-1?:0);

}

**EvalOfPostfixExpr:**

/\* Viraj Ghogale

FY - MCA 1418

Unit 5: Stacks

Program: Evaluation Of A Postfix Expression

\*/

#include"Stack\_09\_11\_2023.cpp"

int main()

{

char postfix[20]; Stack s; int i=0;

float result, n2, n1;

cout << "\*\*\*\*\*Evaluation Of A Postfix Expression\*\*\*\*\* \n\n";

cout << "Enter a postfix expression: \n\n"; gets(postfix);

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

{

if(postfix[i]==' ')

{ i++; continue;

}

if(isdigit(postfix[i]))

{ float x; x = postfix[i] - '48.0'; s.Push(x);

}

else { n2 = s.Pop(); n1 = s.Pop();

switch(postfix[i])

{

case '+': result = n1 + n2;

break;

case '-':

result = n1 - n2;

break;

case '\*':

result = n1\*n2;

break;

case '/':

result = n1/n2;

break;

} // END OF SWITCH

s.Push(result);

} // END OF IF-ELSE OPERATOR

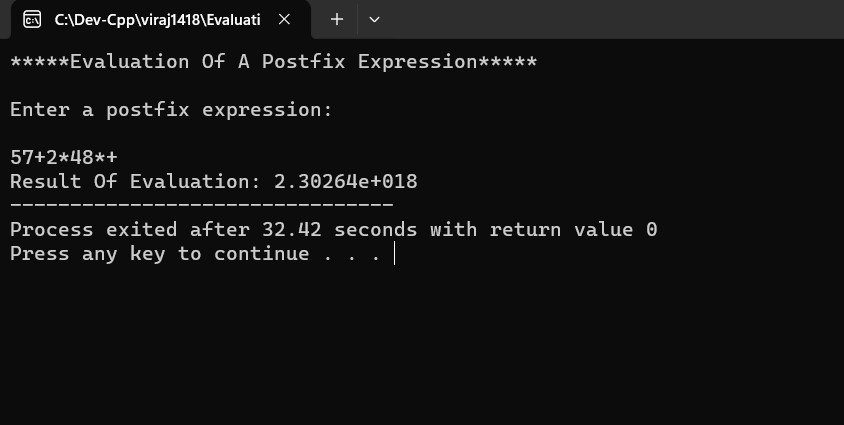
i++;

}// END OF WHILE

cout << "Result Of Evaluation: " <<s.Pop();

}// END OF MAIN

**Output:**



|  |  |
| --- | --- |
| **Program No:** | **15** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 3: Stacks – Balancing of Parenthesis using Stacks** |
| **Objective :** | **Balancing of Parenthesis using Stacks** |

**Source Code:**

/\*

Name:Viraj Ghogale

RollNo:1418

Unit 3: Stacks

Program: Balancing of parenthesis

\*/

#include<iostream>

#include<conio.h>

#define SIZE 20

using namespace std;

// CREATE THE STACK

class Stack

{

char A[SIZE];

int tos;

public:

Stack()

{

tos = -1;

}

void Push(char x);

char Pop();

int Full();

int Empty();

};

void Stack :: Push(char x)

{

if(Full())

{

cout<< "Stack Overflow! ";

return;

}

tos++;

A[tos ]=x;

}

char Stack :: Pop()

{

if(Empty())

{

cout << "St";

return -1;

}

char tmp=A[tos];

tos--;

return tmp;

}

int Stack :: Full()

{

if (tos == SIZE -1)

{

return 1;

}

else

{

return 0;

}

}

int Stack :: Empty()

{

if (tos == -1)

{

return 1;

}

else

{

return 0;

}

}

int main()

{

char expr[20];

char tmp;

int i=0, flag=0;

Stack s;

cout<< " \*\*\* Balancing of paranthesis using a Stack \*\*\*\n\n";

cout<< "Enter an expression: ";

cin>> expr;

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

{

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

{

s.Push(expr[i]);

}

if(expr[i]==']' || expr[i] == ')' || expr [i] == '}')

{

if (s.Empty()) //when extra close bracket

{

flag=1;

break;

}//end if stack empty

tmp = s.Pop();

if ( (tmp == '[' && expr[i] == ']') ||

(tmp == '(' && expr[i] == ')') ||

(tmp == '{' && expr[i] == '}') )

{

i++;

continue;

}//end of matching bracket

else

{

flag =1;

break;

}//end of brackets do not match

}//end of if close bracket

i++;

}//end of while

if(s.Empty() && flag == 0)

{

cout << " Expression is balanced";

}

else

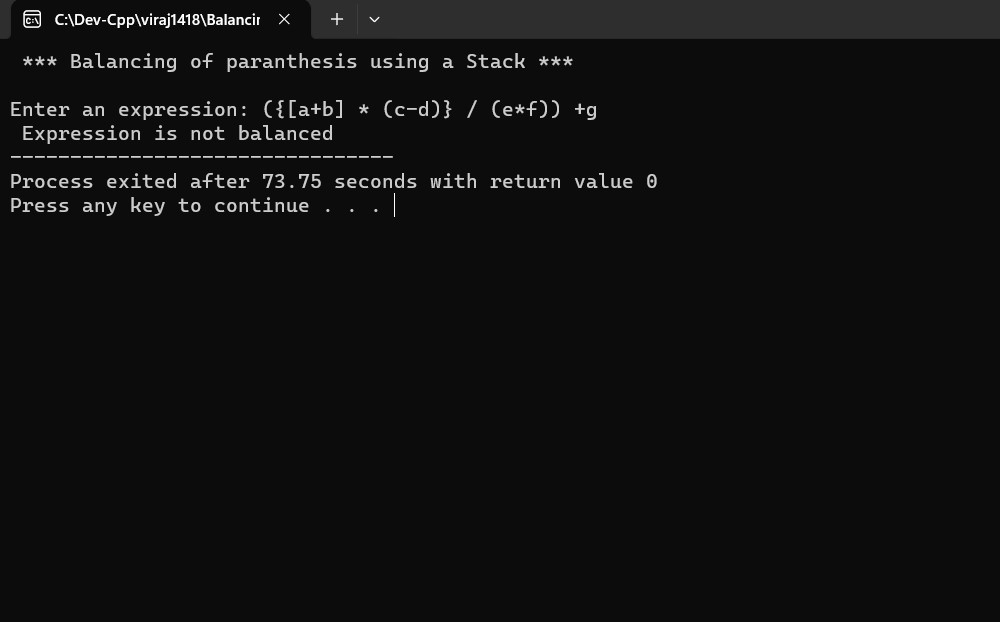
{

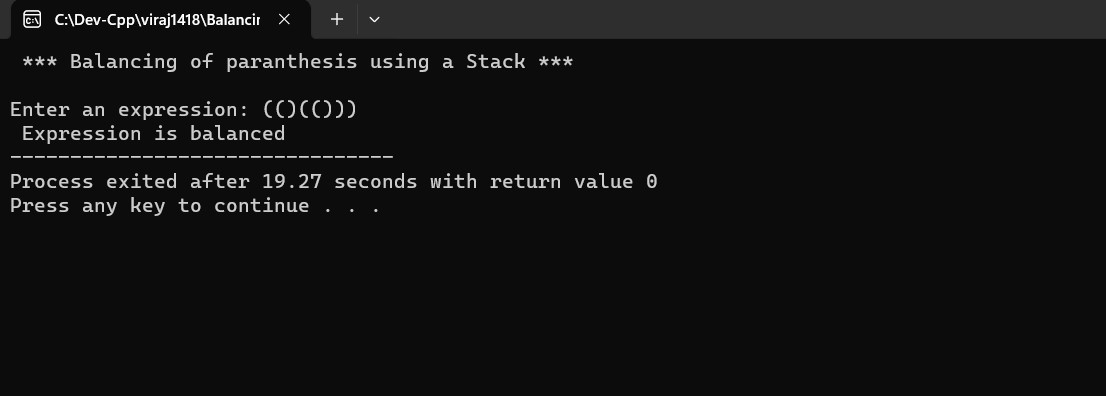
cout <<" Expression is not balanced";

}

}//end of main

**Output:**





|  |  |
| --- | --- |
| **Program No:** | **16** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 4: Queues – Ordinary Queue** |
| **Objective :** | **Ordinary Queue** |

**Source Code:**

/\*

Name : Viraj ghgogale

RollNo : 1418

Unit 4 : Queues

Program : Ordinary Queues

\*/

#include<iostream>

#include<conio.h>

#define SIZE 4

using namespace std;

//1.Node Template -Not required

//2.Array Based Template

class AQueue

{

int A[SIZE];

int front;

int rear;

public:

AQueue()

{

front =-1;

rear =-1;

}

void Enqueue(int x);

void Dequeue();

void PeekFront();

void PeekRear();

void Display();

int Full();

int Empty();

};

//3.functions

void AQueue :: Enqueue(int x)

{

if(Full())

{

cout << "Queue Overflow";

return;

}

rear++;

A[rear]=x;

//

if(front == -1)

{

front++;

}

}//end of Enqueue

int AQueue :: Full()

{

return (rear==SIZE-1 ? 1:0);

}//end of full

int AQueue :: Empty()

{

return (front == -1 ? 1: 0);

}//end of empty

void AQueue :: Dequeue ()

{

if(Empty())

{

cout<<"queue Underflow";

return;

}

int tmp = A[front];

if(front == rear)

{

front = rear =-1;

}

else

{

front++;

}

cout<< "Element removed is: "<<tmp;

}

void AQueue :: PeekFront()

{

if(Empty())

{

cout<< "Queue Underslow!!";

return;

}

cout << "Element in the front of the queue:"<< A[front];

}

void AQueue :: PeekRear()

{

if(Empty())

{

cout<< "Queue Underslow!!";

return;

}

cout << "Element in the rear of the queue:"<< A[rear];

}

void AQueue :: Display()

{

if(Empty())

{

cout <<" Queue underflow!";

return;

}

int i;

cout <<" Queue Contains :\n";

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

{

cout << A[i] << " ";

}

}

//4.menu

int main()

{

int num,ch;

AQueue q;

while(1)

{

system("cls");

cout << "\*\*\*\* Ordinary Queue \*\*\*\n\n";

cout<< "1. Enqueue an element \n";

cout<< "2. Dequeue an element\n";

cout<< "3. Peek FRont \n";

cout<< "4. Peek Rear\n";

cout<< "5. Display\n";

cout<< "6. exit\n";

cout<< "Enter your choice\n";

cin>> ch;

switch (ch)

{

case 1:

cout<<"Enqueue";

cin >>num;

q.Enqueue(num);

getch();

break;

case 2:

cout<<"Dequeue";

q.Dequeue();

getch();

break;

case 3:

cout<<"PeekFront";

q.PeekFront();

getch();

break;

case 4:

cout<<"PeekRear";

q.PeekRear();

getch();

break;

case 5:

cout<<"Display";

q.Display();

getch();

break;

case 6:

exit(1);

default:

cout << "Incorrect choice!";

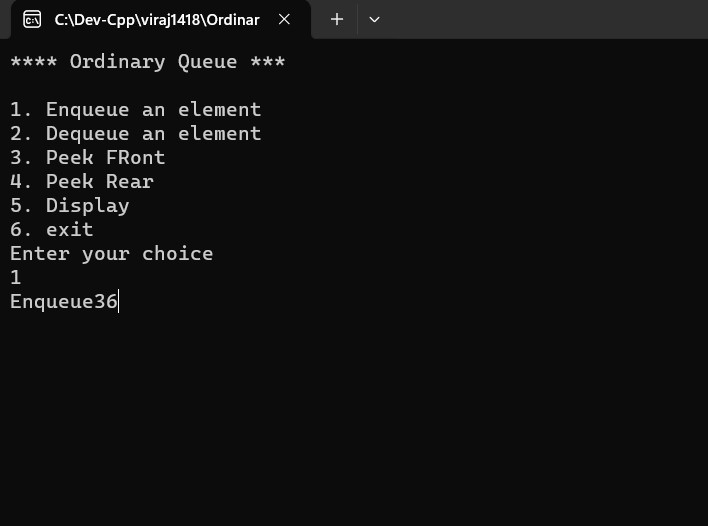
}//end of switch

}//end of while

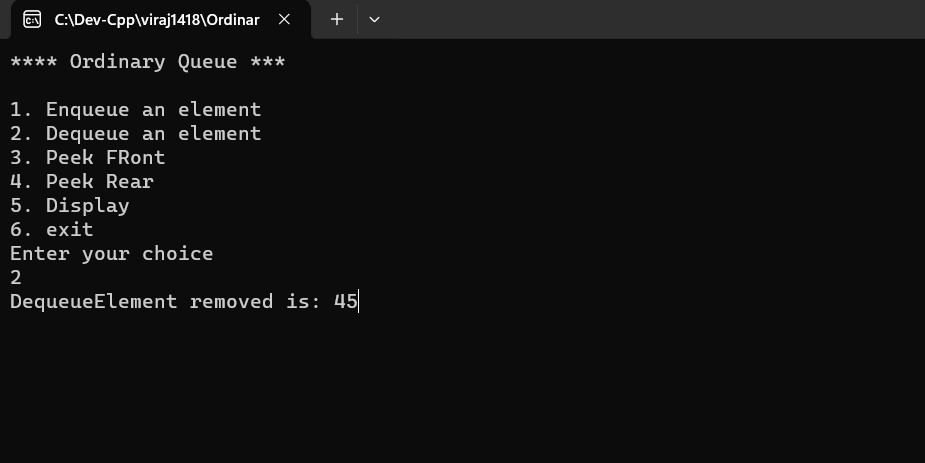
}//end main

**Output:**

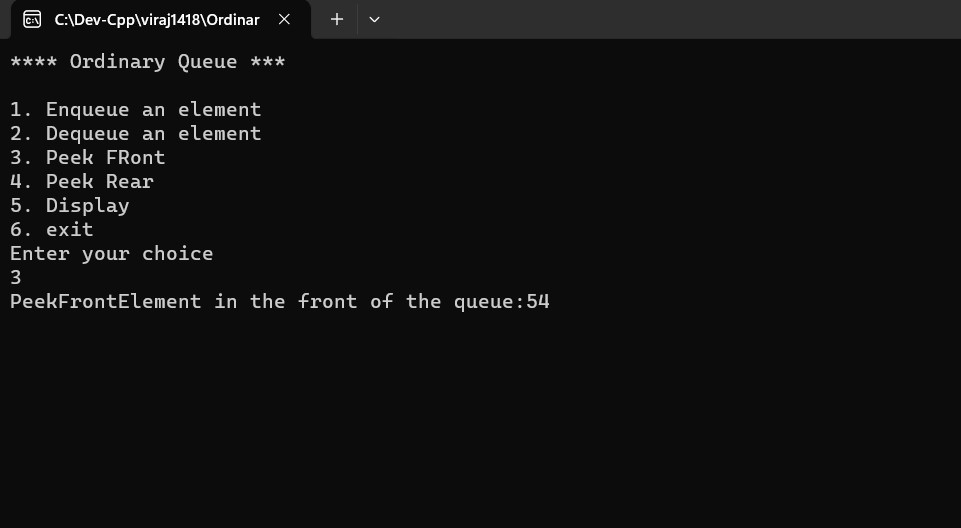
**Enqueue an element –**



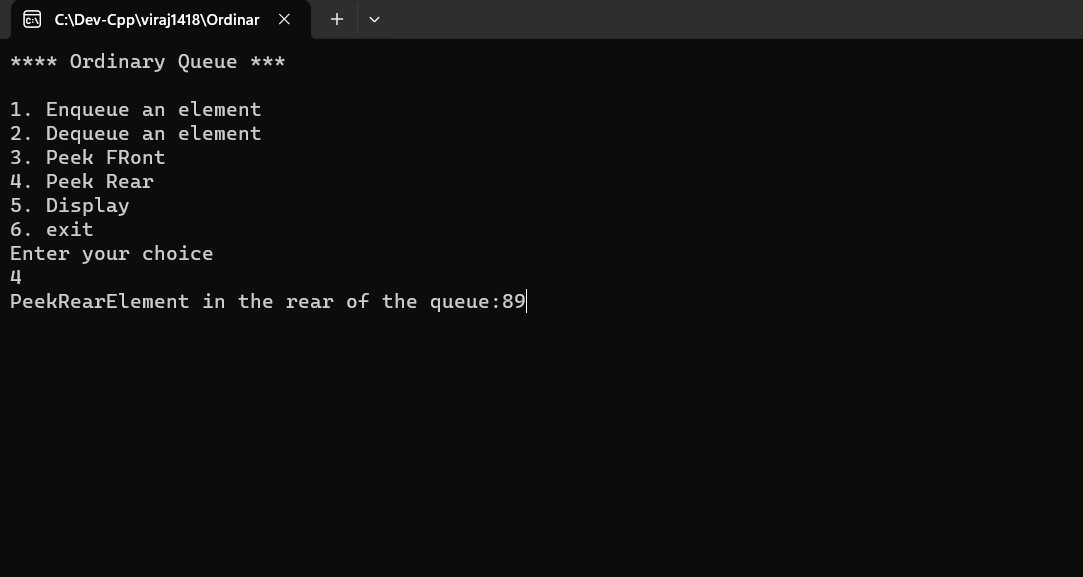
**Dequeue an element –**



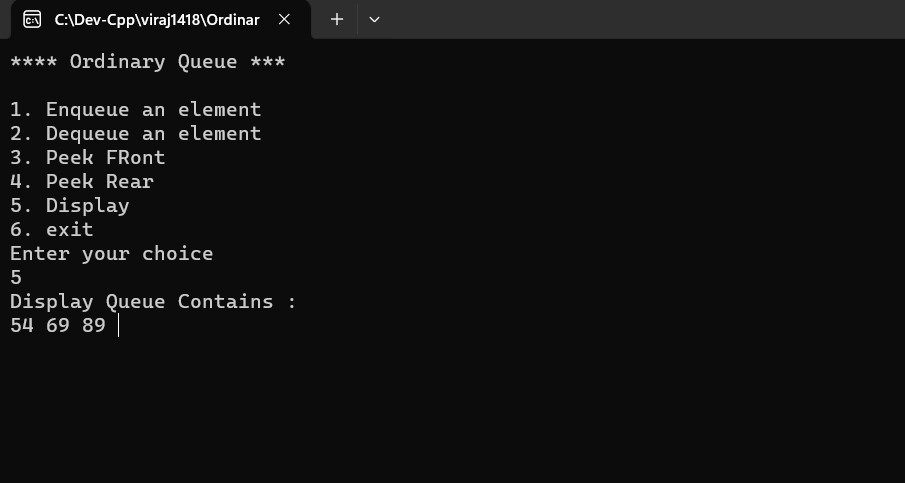
**Peek front –**



**Peek rear –**



**Display the queue –**



|  |  |
| --- | --- |
| **Program No:** | **17** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 4: Queues – Circular Queue** |
| **Objective :** | **Circular Queue** |

**Source Code:**

/\*

Name:Viraj Ghogale

RollNo:1418

Unit 4:

Programm:Circular Queue

\*/

#include<iostream>

#include<conio.h>

#define SIZE 4

using namespace std;

//Node Template

//2.Array

class CQueue

{

int A[SIZE];

int front, rear;

int cnt;

public:

CQueue()

{

front =1;

rear=-1;

cnt=0;

}

void Enqueue(int x);

void Dequeue();

void PeekFront();

void PeekRear();

void Display();

int Full();

int Empty();

};

//3 Functions

int CQueue :: Full()

{

if (cnt == SIZE)

{

return 1;

}

else

{

return 0;

}

}//end of full

int CQueue :: Empty()

{

if (cnt == 0)

{

return 1;

}

else

{

return 0;

}

}//end of empty

void CQueue :: Enqueue(int x)

{

if (Full())

{

cout << "Queue Overflow!";

return;

}

if (rear == SIZE-1)

{

rear = 0;

}

else

{

rear++;

}

A[rear] = x;

if (front == -1)// special case

{

front ++;

}

cnt ++;

}

void CQueue :: Dequeue()

{

if (Empty())

{

cout<< "Queue Underflow !";

return;

}

int tmp = A[front];

if(front == rear)//single element deletion

{

front = -1;

rear = -1;

}

else if (front == SIZE-1)// implement circularity

{

front =0;

}

else

{

front ++;

}

cout << "Element removed :" <<tmp;

cnt--;

}

void CQueue :: PeekFront()

{

if(Empty())

{

cout<< "Queue UNderflow!";

return;

}

cout<< "Element at the front "<< A[front];

}//end of peekfront

void CQueue :: PeekRear()

{

if(Empty())

{

cout<< "Queue Underflow!";

return;

}

cout<< "Element at the rear "<< A[rear];

}//end of peekfront

void CQueue :: Display()

{

if(Empty())

{

cout << "Queue Underflow! ";

return;

}

int i,j;

for(j=1,i=front ; j<= cnt ; j++,i=(i+1)%SIZE)

{

cout << A [i];

}

}

//menu

int main()

{

int ch,num;

CQueue q;

while(1)

{

system("cls");

cout <<"\*\*\* Circular Queue\n\n";

cout <<"1. Enqueue Opertaion\n";

cout <<"2. Dequeue Opertaion\n";

cout <<"3. PeekFront Opertaion\n";

cout <<"4. PeekRear Opertaion\n";

cout <<"5. Display the queue\n";

cout <<"6. Exit\n";

cout <<"Enter your choice: \n";

cin>>ch;

switch(ch)

{

case 1:

cout<<"Enqueue: \n";

cin>>num;

q.Enqueue(num);

getch();

break;

case 2:

cout<<"Dequeue : \n";

q.Dequeue();

getch();

break;

case 3:

q.PeekFront();

getch();

break;

case 4:

q.PeekRear();

getch();

break;

case 5:

q.Display();

getch();

break;

case 6:

exit(1);

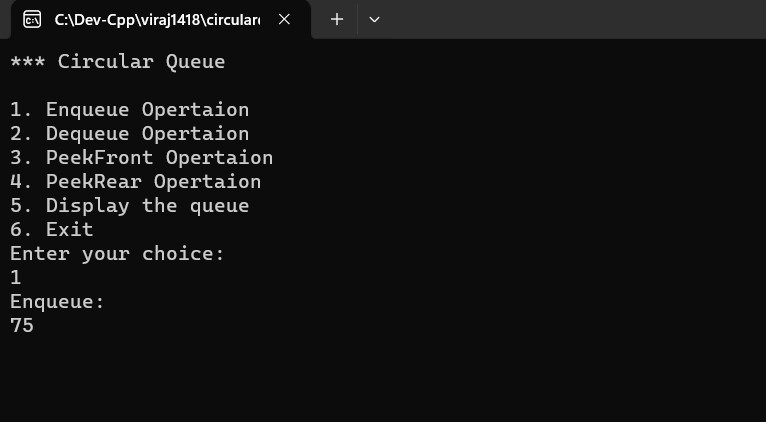
default:

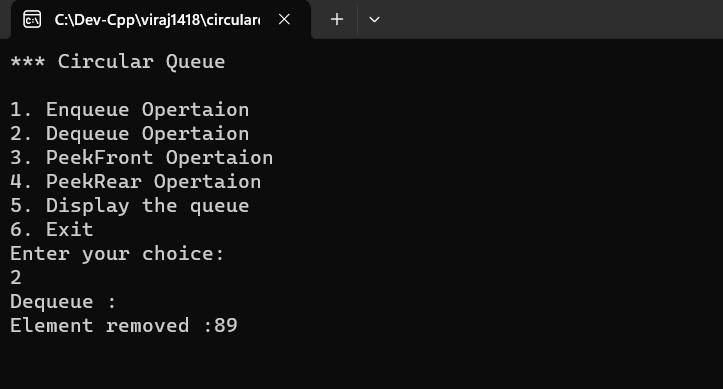
cout<<" Incorrect choice !";

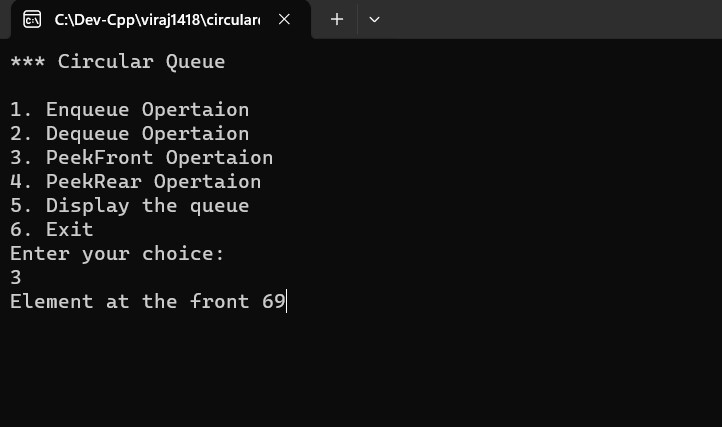
}//end of switch

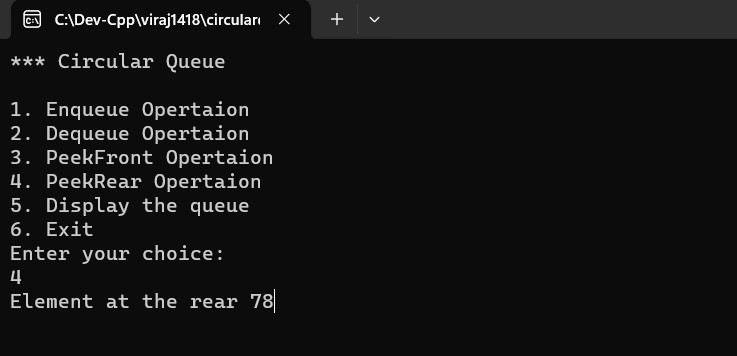
}//end of while

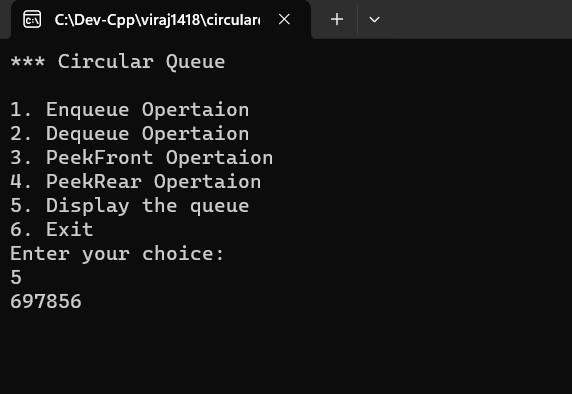
}//end of main

**Output:** 









|  |  |
| --- | --- |
| **Program No:** | **18** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 4: Queues – Priority Queue** |
| **Objective :** | **Priority Queue** |

**Source Code:**

/\*

Name:Viraj Ghogale

RollNo:1418

Unit:

\*/

#include<iostream>

#include<conio.h>

using namespace std;

class PQNode

{

public:

int data;

int priority;

PQNode \*next;

};

//2.List based template

class PQueue

{

PQNode \*front;

PQNode \*rear;

public:

PQueue()

{

front = NULL;

rear = NULL;

}

void Enqueue(int x,int p);

void Dequeue();

void PeekFront();

void PeekRear();

void Display();

int Empty();

};

//3.Functions

void PQueue :: Enqueue(int x ,int p)

{

//1.

PQNode \*t =new PQNode;

t->data=x;

t->priority=p;

t->next = NULL;

//2.1st node in the queue

if(front == NULL)

{

front =t;

rear =t;

return;

}

//3.Traverese

PQNode \*tmp =front;

PQNode \*prev = NULL;

while (tmp!= NULL && tmp->priority < t->priority)

{

prev = tmp;

tmp =tmp->next;

}

//4.

if(tmp==front)// front node insertion

{

t->next =front;

front =t;

}

else if (tmp == NULL)

{

rear->next =t;

rear =t;

}

else //any other node

{

prev->next =t;

t->next =tmp;

}

}//end of enqueue

void PQueue :: Display()

{

if(Empty())

{

cout << "Queue Underflow!";

return;

}

PQNode \*tmp = front;

cout << endl;

while(tmp!=NULL)

{

cout <<"Data: " << tmp->data <<" Priority: " <<tmp->priority << endl;

tmp = tmp->next;

}

}

int PQueue :: Empty()

{

if(front == NULL)

{

return 1;

}

else

{

return 0;

}

}

void PQueue :: PeekRear()

{

if(Empty ())

{

cout << "Queue UnderFlow!";

return;

}

cout <<"Data: " <<front->data << " Priority: " <<front->priority;

}

void PQueue :: PeekFront()

{

if(Empty ())

{

cout << "Queue UnderFlow!";

return;

}

cout <<"Data: " <<rear->data << " Priority: " <<rear->priority;

}//

void PQueue :: Dequeue()

{

if(Empty())

{

cout << " Queue Underflow!";

return;

}

PQNode \*tmp = front;

if ( front == rear)

{

front = NULL;

rear = NULL;

}

else

{

front = front->next;

}

cout << " Element removed: " << tmp->data << " with priority: " <<tmp->priority;

delete tmp;

}

//4.Menu

int main()

{

PQueue q;

int ch, num,pr;

while (1)

{

system("cls");

cout<<"\*\*\*\*PriorityQueue Menu\*\*\*\*\n\n";

cout<<"1. Enqueue a node\n";

cout<<"2. Dequeue a node\n";

cout<<"3. PeekFront\n";

cout<<"4. PeekRear\n";

cout<<"5. Display \n";

cout<<"6. Exit\n";

cout<<" Enter Your Choice: ";

cin>> ch;

switch(ch)

{

case 1: cout<<"Enter an element : ";

cin>>num;

cout<< "Enter the priority: ";

cin>>pr;

q.Enqueue(num,pr);

getch();

break;

case 2: cout<<"Dequeue ";

q.Dequeue();

getch();

break;

case 3: cout<<"PeekFront\n";

q.PeekFront();

getch();

break;

case 4: cout<<"PeekRear\n";

q.PeekRear();

getch();

break;

case 5: cout<<"Display";

q.Display();

getch();

break;

case 6:

exit(1);

default:

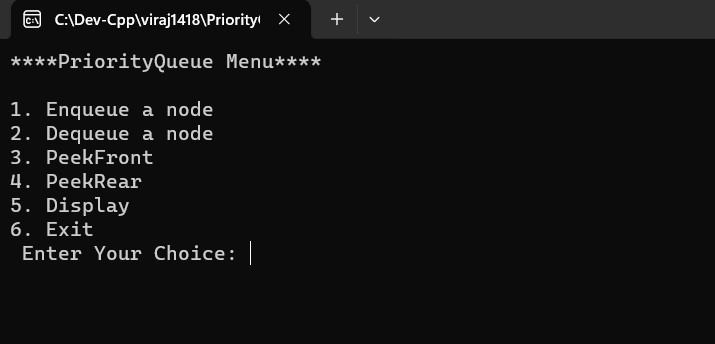
cout<<"Incorrect Choice";

getch();

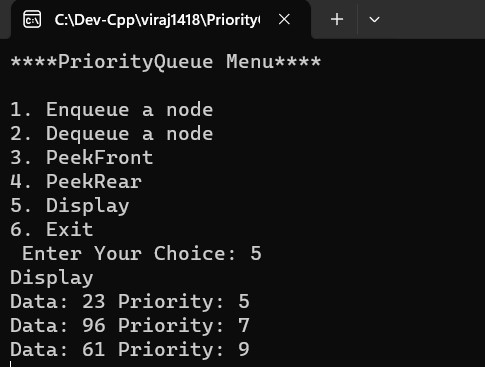
}//end of switch

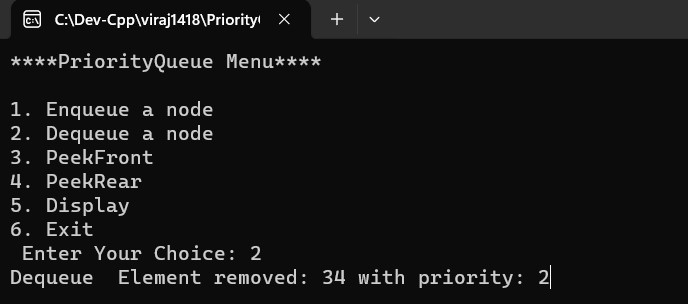
}//end of while

}//end of main menu

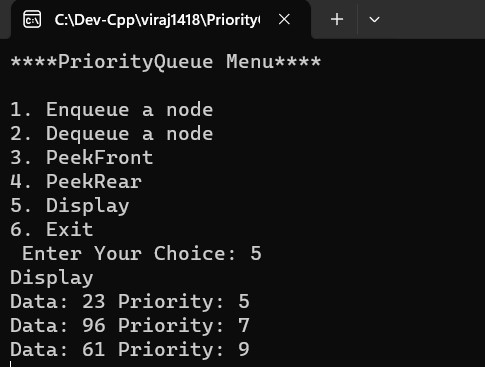
**Output:** 

**Enqueue an element –**



**Dequeue an element –** 

**Display the queue –**



|  |  |
| --- | --- |
| **Program No:** | **19** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 4: Queues – Double Ended Queue** |
| **Objective :** | **Double Ended Queue** |

**Source Code:**

/\*

Name: Viraj Ghogale

Roll No: 1418

Unit-4 : Queues

Program: Double Ended Queue \*/

#include<iostream>

#include<conio.h>

using namespace std;

/\* 1. Node Template - NOT REQUIRED \*/

class DQNode

{

public:

int data;

DQNode \*right;

DQNode \*left;

};

/\* 2. Array Based Template \*/

class DQueue

{

DQNode \*front;

DQNode \*rear;

DQNode \*tmp;

public: DQueue()

{

front = NULL;

rear = NULL;

}

void EnqueueFront(int x);

void EnqueueRear(int x);

void DequeueFront();

void DequeueRear();

void PeekFront();

void PeekRear();

void Display();

};

/\* 3. Functions \*/

void DQueue :: EnqueueFront(int x)

{

DQNode \*t = new DQNode;

t->data=x;

t->right = NULL;

t->left = NULL;

if(front == NULL)

{

front = t;

rear = t;

}

else

{

t->right = front;

front->left=t;

front = t;

}

}

void DQueue :: EnqueueRear(int x)

{

DQNode \*t = new DQNode;

t->data = x;

t->left = NULL;

t->right = NULL;

if(front == NULL)

{

front = t;

rear = t;

}

else

{

rear->right = t;

t->left = rear;

rear = t;

}

}

void DQueue :: DequeueFront()

{

if(front == NULL)

{

cout << "Queue Underflow!";

return;

}

if(front == rear)

{

front = NULL;

rear = NULL;

}

else

{

front = front->right;

front->left = NULL;

}

delete tmp;

}

void DQueue :: DequeueRear()

{

if(front == NULL)

{

cout << "Queue Underflow!";

return;

}

DQNode \*tmp = front;

if(front == rear)

{

front = NULL;

rear = NULL;

}

else

{

rear = rear->left;

rear->right = NULL;

}

delete tmp;

}

void DQueue :: PeekFront()

{

if(front == NULL)

{

cout << "Queue Underflow!";

return;

}

DQNode \*tmp = front;

cout << "Element at front: " <<tmp->data;

}

void DQueue :: PeekRear()

{

if(rear == NULL)

{

cout << "Queue Underflow!";

return;

}

DQNode \*tmp = rear;

cout << "Element at front: " <<tmp->data;

}

void DQueue::Display()

{

DQNode \*tmp = front;

if (tmp == NULL)

{

cout << "Empty Queue";

return;

}

else

{

cout << "Queue elements: ";

while (tmp != NULL)

{

cout << tmp->data << " ";

tmp = tmp->right;

}

cout << endl;

}

}

/\* 4. Menu \*/

int main()

{

int ch,num;

DQueue q;

while(1)

{

system("cls");

cout << " \*\*\* Double-Ended Queue\*\*\*\n" <<endl;

cout << "1. Enqueue Front\n";

cout << "2. Enqueue Rear\n";

cout << "3. Dequeue Front\n";

cout << "4. Dequeue Rear\n";

cout << "5. Peek Front\n";

cout << "6. Peek Rear\n";

cout << "7. Display the queue\n";

cout << "8. Exit\n\n";

cout << "Enter your choice: \n";

cin>>ch;

switch(ch)

{

case 1:

cout << "Enter An Element: ";

cin >> num;

q.EnqueueFront(num);

getch();

break;

case 2:

cout << "Enter An Element: ";

cin >> num;

q.EnqueueRear(num);

getch();

break;

case 3:

cout << "Dequeued element from front ";

q.DequeueFront();

getch();

break;

case 4:

cout << "Dequeued element from rear";

q.DequeueRear();

getch();

break;

case 5:

cout << "Peekfront\n";

q.PeekFront();

getch();

break;

case 6:

cout << "Peekrear\n";

q.PeekRear();

getch();

break;

case 7:

cout << "The queue is\n";

q.Display();

getch();

break;

case 8:

exit(1);

default:

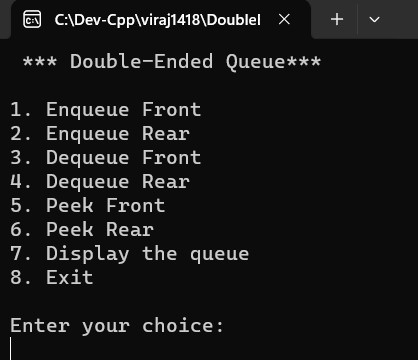
cout << "Invalid Option";

}//End of Switch

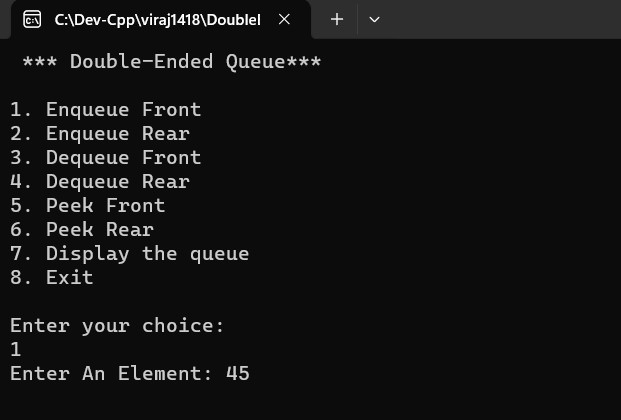
}//End of While

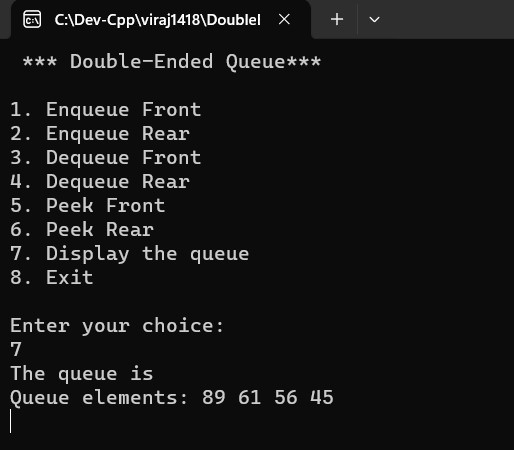
}//End of Main

**Output:**

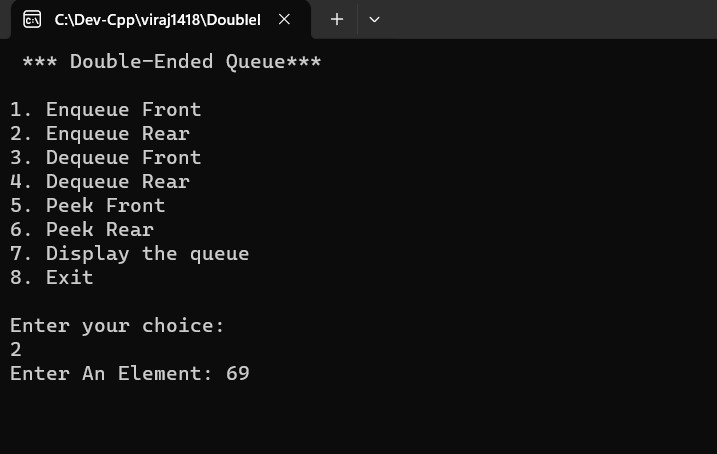


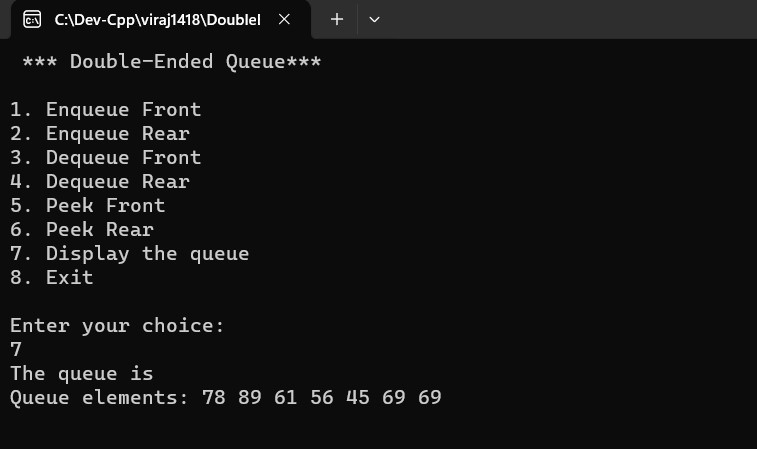
**Enqueue front –**



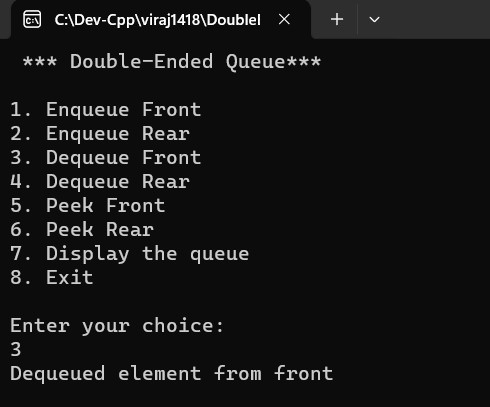


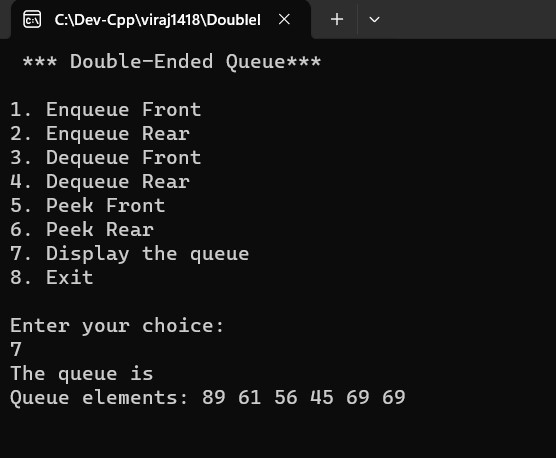
**Enqueue rear –**



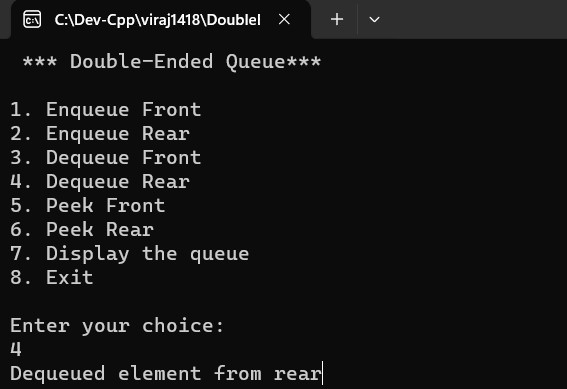


**Dequeue front –**

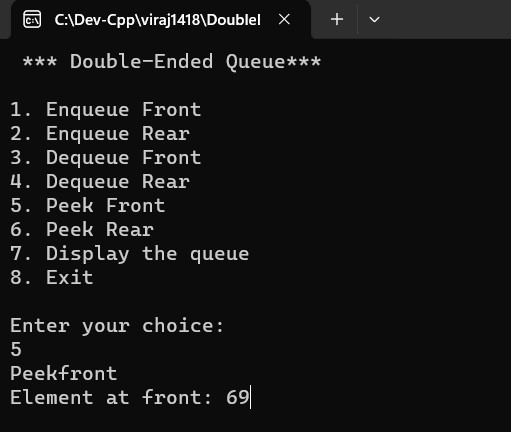




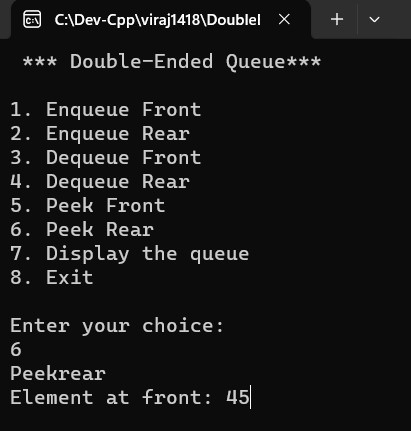
**Dequeue Rear:**



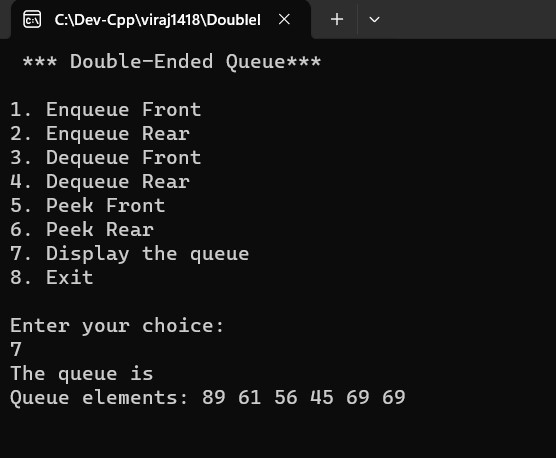
**Peek Front:**



**Peek Rear:**



**Display:**



|  |  |
| --- | --- |
| **Program No:** | **20** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 5: Linked Lists – Singly Linked Lists** |
| **Objective :** | **Singly Linked Lists** |

**Source Code:**

/\*Name :Viraj Ghogale

Roll No:1418

Unit 5: Linked list

Program: Singly linked list

\*/

#include<iostream>

#include<conio.h>

using namespace std;

/\*1. Node Template\*/

class Node

{

public:

int data;

Node \*next;

};

/\* 2. List Template\*/

class List

{

Node \*head;

public:

List()

{

head = NULL;

}

void Insert(int x);

void Del(int x);

void Search(int x);

void Length();

void Display();

};

/\*3. Functions\*/

void List :: Insert(int x)

{

//1. Make a new node t

Node \*t = new Node; // here it is creating a memory address via t

t->data = x;

t->next = NULL;

//2. Special case: First node in the SLL

if(head == NULL)

{

head = t;

}

else //3. Traverse till last node and attach t

{

Node \*temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = t;

}

}// end of insert

void List :: Display()

{

Node \*temp = head;

while(temp!= NULL)

{

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

temp = temp->next;

}

cout<<"End of the list";

}//end of display

void List :: Length()

{

Node \*tmp = head;

int cnt =0;

while(tmp!=NULL)

{

cnt++;

tmp = tmp->next;

}

cout << "Length of the list: " << cnt;

}//end of length

void List :: Search(int x)

{

Node \*tmp = head;

int flag = 0;

while(tmp!=NULL)

{

if(tmp->data == x)

{

flag =1;

break;

}

tmp =tmp->next;

}

if(flag ==1)

{

cout << x <<" is found";

}

else

{

cout << x << "is not found";

}

}// end of search

void List :: Del(int x)

{

//1. Empty List - RETURN CONTROL

if(head == NULL)

{

cout << "Empty list!";

return;

}

//2.Search for x

Node \* tmp =head;

Node \* prev =NULL;

int flag=0;

while (tmp != NULL)

{

if(tmp->data == x)

{

flag= 1;

break;

}

prev = tmp;

tmp=tmp->next;

}

//3. Unsuccessful Search - Return Control

if (flag == 0)

{

cout << x << " is not found!";

return;

}

//4. Successful Search

//a Single node deletion

if(tmp == head && tmp->next == NULL)

{

head = NULL;

}

else if(tmp == head )//b.head node deletion

{

head = tmp->next;

}

else if(tmp->next == NULL) //c. tail end deletion

{

prev->next =NULL;

}

else //d. any other node

{

prev->next = tmp->next;

}

//5.delete memory

delete tmp;

}//end of del

/\*4. Main Menu\*/

int main()

{

List SinglyLinkedList;

int choice, number;

while(1)

{

system("cls");

cout<<"\*\*\* Singly Linked List\*\*\*\n\n";

cout<<"1. Insert a Node\n";

cout<<"2. Delete a Node"<<endl;

cout<<"3. Search for a Node"<<endl;

cout<<"4. Length of the list"<<endl;

cout<<"5. Display the list"<<endl;

cout<<"6. Exit" <<endl<<endl;

cout<<"Enter Your Choice:";

cin>> choice;

switch(choice)

{

case 1:

// cout<<"Insert";

// getch();

cout<<"Insert a Value:";

cin>>number;

SinglyLinkedList.Insert(number);

getch();

break;

case 2:

cout<<"Enter a value to Delete";

cin >> number;

SinglyLinkedList.Del(number);

getch();

break;

case 3:

cout<<"Enter a value";

cin>>number;

SinglyLinkedList.Search(number);

getch();

break;

case 4:

cout<<"Lenght";

getch();

SinglyLinkedList.Length();

getch();

break;

case 5:

cout<<"Display";

SinglyLinkedList.Display();

getch();

break;

case 6:

exit(1);

default:

cout<<"Incorrect Choice";

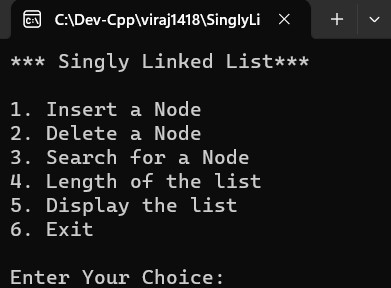
getch();

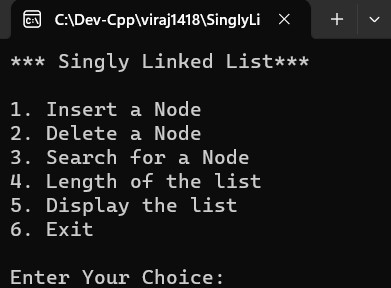
}//end of switch

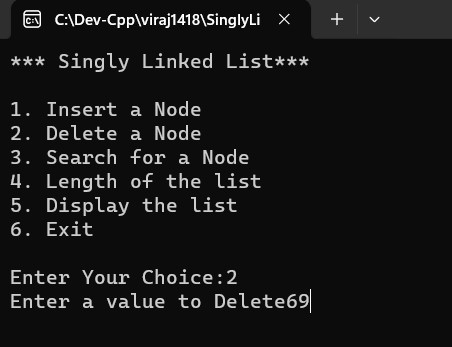
}//end of while

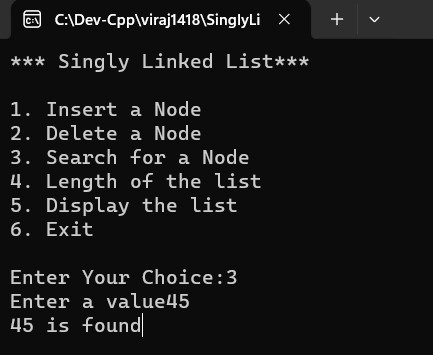
}//end of main

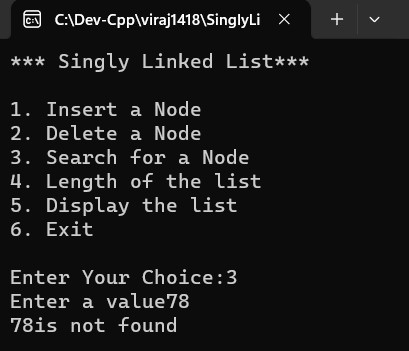
**Output:**

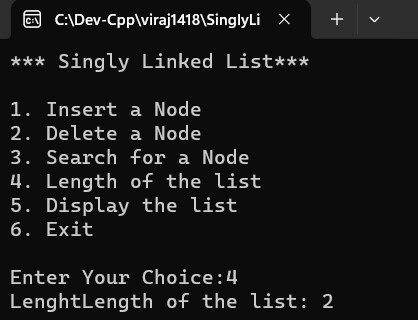


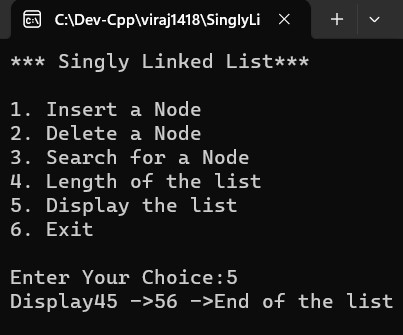


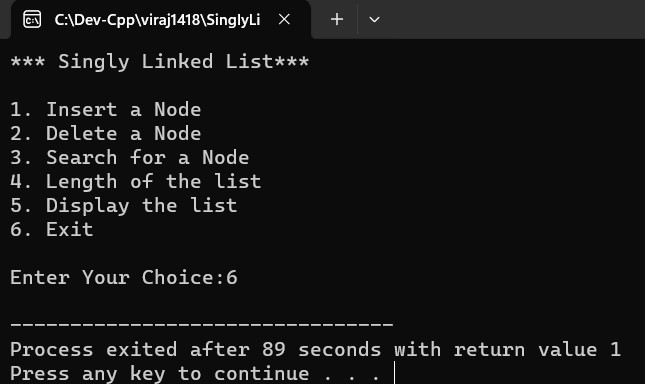












|  |  |
| --- | --- |
| **Program No:** | **21** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 5: Linked Lists – Circular Linked Lists** |
| **Objective :** | **Circular Linked Lists** |

**Source Code:**

/\*

name: viraj

roll no:1418

unit : 5 Linked List

program: circularlinkedlist

\*/

#include<iostream>

#include<conio.h>

using namespace std;

/\*1. Node Template\*/

class CNode

{

public:

int data;

CNode \*next;

};

/\*2. List Template\*/

class CList

{

CNode \*first;

CNode \*last;

public:

CList()

{

first = last = NULL;

}

void Insert(int x);

void Delete(int x);

void Search(int x);

void Length();

void Display();

};

/\*3.Functions\*/

void CList :: Insert(int x)

{

// MAke a new node t

CNode \*t =new CNode;

t->data = x;

t->next =NULL;

//First node inn the CLl

if(first == NULL)

{

first =t;

last =t;

last->next = first;

}

else //Attach t to the last node

{

last->next = t; //Link last new node to t

last = t; //shift last to t

last->next =first; //Establish circularity

}

} //end of insert

void CList :: Display()

{

CNode \*tmp =first;

if(first == NULL)

{

cout << "Empty CLL!";

return;

}

do

{

cout << tmp->data << "->";

tmp= tmp->next;

}

while(tmp != first);

}//end of display

void CList ::Length()

{

if(first == NULL)

{

cout << "Empty CLL!";

return;

}

CNode \*tmp =first;

int cnt =0;

do

{

cout << tmp->data << "->";

tmp= tmp->next;

}

while(tmp != first);

cout << "Number of nodes: " << cnt;

}//end of length

void CList :: Search (int x)

{

if (first == NULL)

{

cout << "Empty Search" ;

return;

}

CNode \*tmp =first;

int flag =0;

do

{

if(tmp->data == x)

{

flag =1;

break;

}

tmp =tmp->next;

}

while(tmp!=first);

if(flag ==1)

{

cout <<x <<"is found";

}

else

{

cout << x << "is not found";

}

}//end of search

void CList:: Delete(int x)

{

if( first == NULL)

{

cout << "Empty CLL!";

return ;

}

//2. Search for x

CNode \*tmp =first;

CNode \*prev =NULL;

int flag =0;

do

{

if(tmp->data ==x)

{

flag=1;

break;

}

prev =tmp;

tmp= tmp->next;

}

while(tmp!=first);

//3. Unsuccessful Search -RETURN CONTROl

if(flag ==0)

{

cout << "Unsuccessful Search";

return;

}

//4. Successfulm search

//4a. Single Node Deletion

if(tmp ==first && tmp == last)

{

first = NULL;

last =NULL;

}

else if (tmp == first)

{

first = first->next;

last->next =first;

}

else if (tmp ==last)

{

last = prev;

last->next = first;

}

else

{

prev->next =tmp->next;

}

//5. Delete memory occupied by x - tmp

delete tmp;

}//end of Del

/\*Main Menu\*/

int main()

{

CList c;

int choice, number;

while (1)

{

system("cls");

cout<<"\*\*\*\*Circular Linked List Menu\*\*\*\*\n\n";

cout<<"1. Insert a node in the CLL\n\n";

cout<<"2. Delete a node from the CLL\n\n";

cout<<"3. Search for a node in the CLL\n\n";

cout<<"4. Length of the CLL\n\n";

cout<<"5. Display the CLL\n\n";

cout<<"6. Exit\n\n";

cout<<" Enter Your Choice: ";

cin>> choice;

switch(choice)

{

case 1: cout<<"Insert option ";

cin>>number;

c.Insert(number);

getch();

break;

case 2: cout<<"Enter a value: ";

cin>> number;

c.Delete(number);

c.Display();

getch();

break;

case 3: cout<<"Enter the value: ";

cin >> number;

c.Search(number);

getch();

break;

case 4: cout<<"The length is";

c.Length();

getch();

break;

case 5:

c.Display();

getch();

break;

case 6:

exit(1);

default:

cout<<"Incorrect Choice";

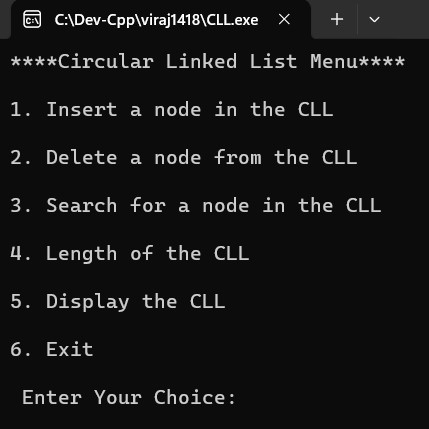
getch();

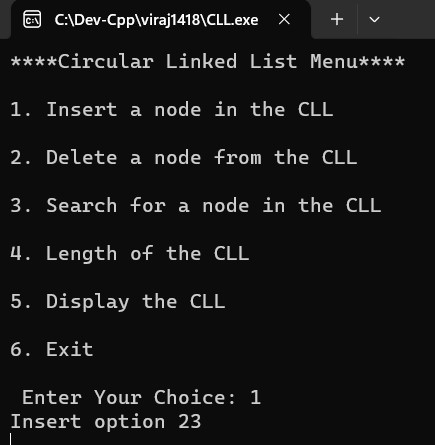
}//end of switch

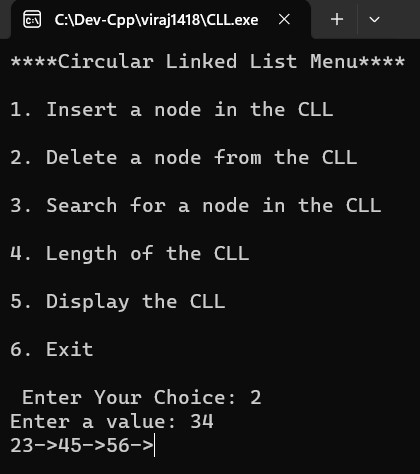
}//end of while

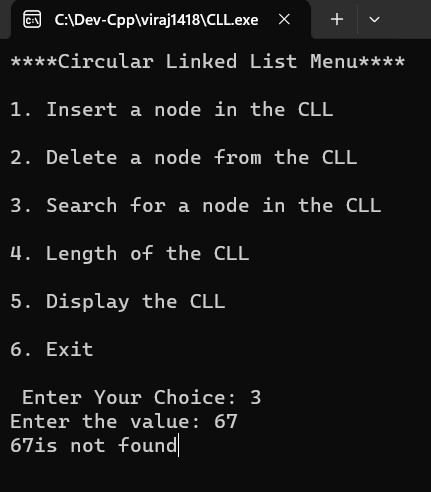
}//end of main menu

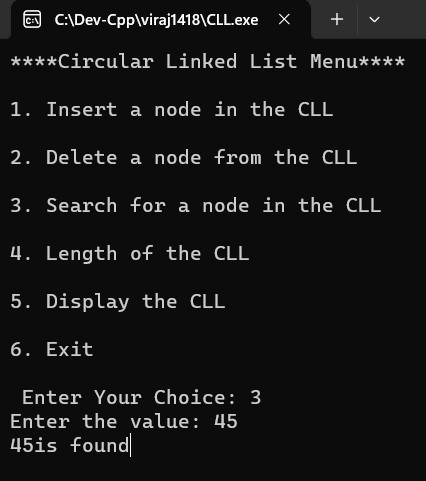
**Output:**

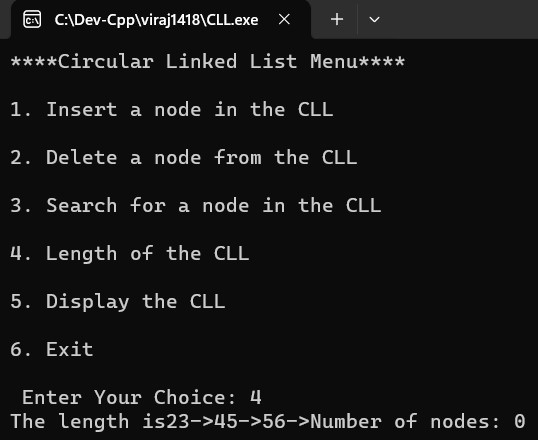


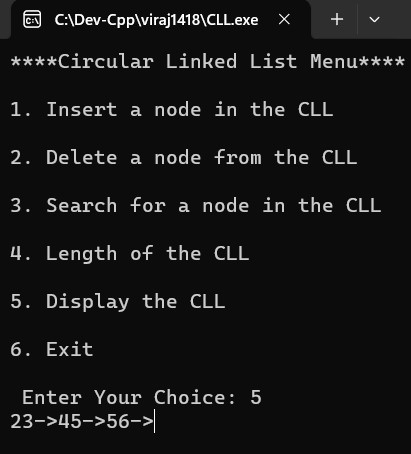












|  |  |
| --- | --- |
| **Program No:** | **22** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 5: Linked Lists – Doubly Linked Lists** |
| **Objective :** | **Doubly Linked Lists** |

**Source Code:**

/\*

Name : Viraj Ghogale

Roll No:1418

Unit:

Programm :Doubly linked List

\*/

#include<iostream>

#include<conio.h>

using namespace std;

/\*1. Node Template\*/

class DNode

{

public:

DNode \*left;

int data;

DNode \*right;

};

/\* 2. list Template\*/

class DList

{

DNode \*head;

DNode \*tail;

public:

DList()

{

head =tail =NULL;

}

void Insert(int x);

void Delete(int x);

void Search(int x);

void Length();

void Display();

};

/\* Functions \*/

void DList :: Insert(int x)

{

//make a new nodw t

DNode \*t =new DNode;

t->data =x;

t->right =NULL;

t->left =NULL;

//First node in the DLL

if(head == NULL)

{

head =t;

tail=t;

}

else

{

tail ->right =t; //attach t to right of the tail

t->left = tail; //attach t to left of the tail

tail =t; //shift the tail

}

}// end of insert

void DList :: Display()

{

DNode \*tmp =head;

cout << "DLL in forward direction :";

while(tmp!= NULL)

{

cout << tmp->data << "<->";

tmp = tmp->right;

}

cout << "NULL \n\n";

tmp =tail;

cout << "DLL inn reverse direction: ";

while(tmp!= NULL)

{

cout << tmp->data << "<->";

tmp = tmp->left;

}

cout << "NULL \n\n";

}//end of display

void DList :: Length()

{

DNode \*tmp = head;

int cnt;

while( tmp!= NULL)

{

cnt++ ;

tmp = tmp->right;

}

cout << "The length is :"<<cnt;

}// end of length

void DList :: Search(int x)

{

DNode \*tmp=head;

int flag =0;

while(tmp != NULL)

{

if(tmp-> data == x)

{

flag=1;

break;

}

tmp =tmp->right;

}

if (flag == 1)

{

cout <<x<< " is found ";

}

else

{

cout <<x<< " not found! ";

}

}//end of search

void DList :: Delete(int x){

//Empty List - RETURN CONTROL

if(head==NULL)

{

cout << "Empty DLL";

return;

}

//Search for x

DNode \*tmp = head;

DNode \*prev = NULL;

int flag = 0;

while(tmp!=NULL)

{

if(tmp->data==x)

{

flag = 1;

break;

}

prev = tmp;

tmp = tmp->right;

}

//Unsuccessful search - return control

if(flag == 0)

{

cout << x << " is not found!";

return;

}

//4.Successfull search

//4a.single node deletion

if(tmp==head && tmp==tail)

{

head = NULL;

tail = NULL;

}

//4b.Head Node Deletion

else if(tmp==head)

{

head = head->right;

head->left = NULL;

}

//4c.Tail Node Deletion

else if(tmp==tail)

{

tail = prev;

tail->right = NULL;

}

//4d.Any other Node Deletion (NOT head and NOT tail)

else

{

prev->right = tmp->right;

(tmp->right)->left = prev;

}

//delete tmp

delete tmp;

}

//END OF DELETE FUNCTION

/\*Main Menu\*/

int main()

{

DList d;

int choice, number;

while (1)

{

system("cls");

cout<<"\*\*\*\*Doubly Linked List Menu\*\*\*\*\n\n";

cout<<"1. Insert a node in the DLL\n\n";

cout<<"2. Delete a node from the DLL\n\n";

cout<<"3. Search for a node in the DLL\n\n";

cout<<"4. Length of the DLL\n\n";

cout<<"5. Display the DLL\n\n";

cout<<"6. Exit\n\n";

cout<<" Enter Your Choice: ";

cin>> choice;

switch(choice)

{

case 1: cout<<"Insert option ";

cin>>number;

d.Insert(number);

getch();

break;

case 2: cout<<"Enter a value: ";

cin>> number;

d.Delete(number);

d.Display();

getch();

break;

case 3: cout<<"Enter the value: ";

cin >> number;

d.Search(number);

getch();

break;

case 4: cout<<"The length is";

d.Length();

getch();

break;

case 5:

d.Display();

getch();

break;

case 6:

exit(1);

default:

cout<<"Incorrect Choice";

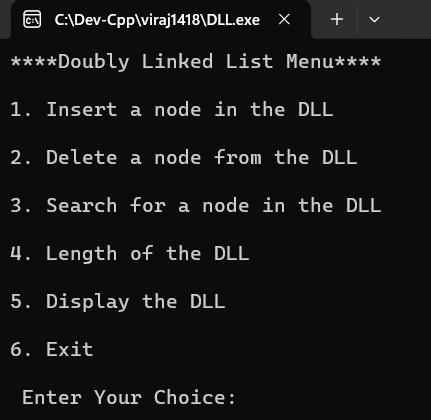
getch();

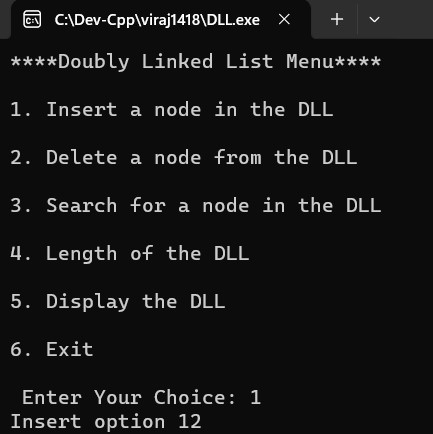
}//end of switch

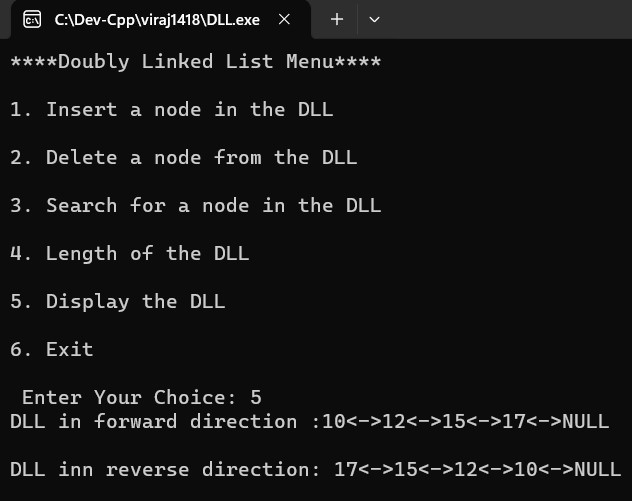
}//end of while

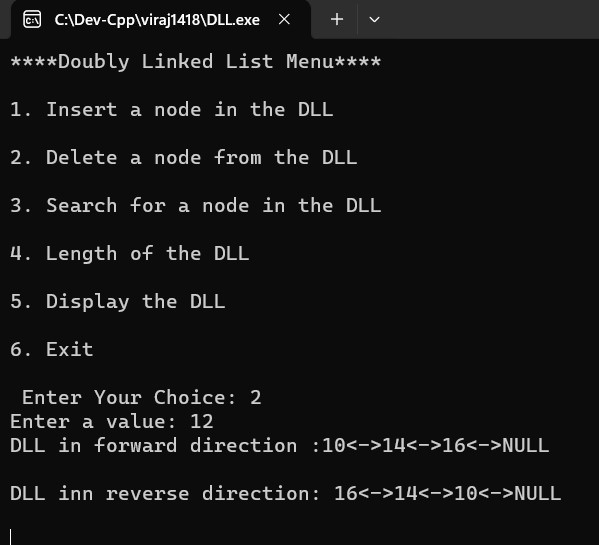
}//end of main menu

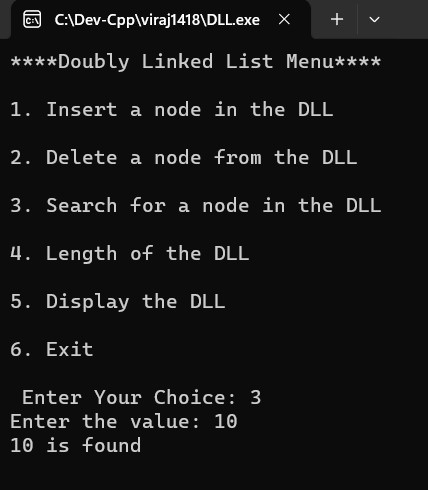
**Output:**

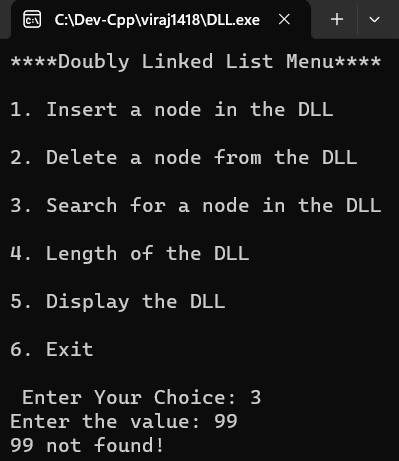


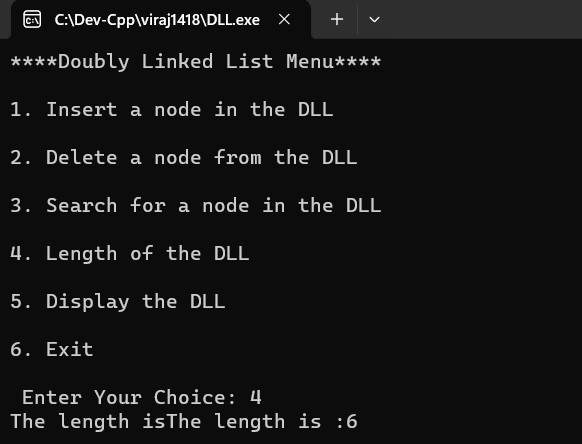


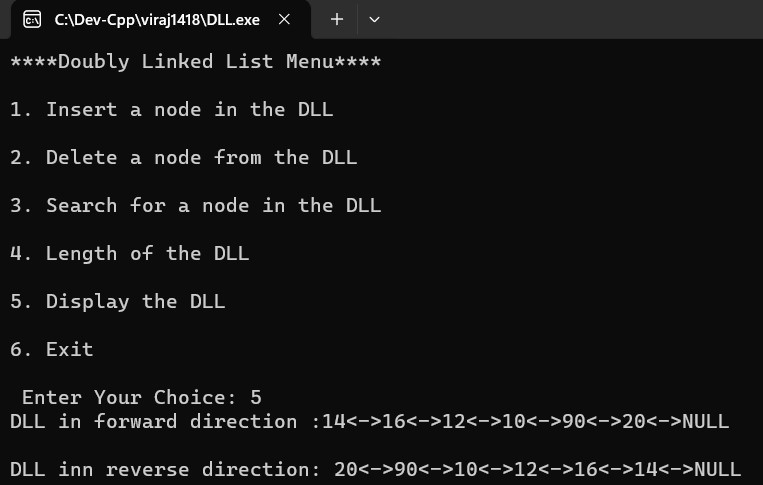












|  |  |
| --- | --- |
| **Program No:** | **23** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 4: Queues – Polynomial Addition using Lists** |
| **Objective :** | **Polynomial Addition using Lists** |

**Source Code:**

/\*

Name: Viraj Ghogale

FYMCA - 1418

Unit 5: lists

Program: Addition Of Polynomials Using Lists

\*/

/\* Name :

RollNo:1418

Unit 5: Lists

Program:Addition of Polynomial

\*/

#include<iostream>

#include<conio.h>

using namespace std;

//Node TEmplate

class PNode

{

public:

int Coeff;

int Exp;

PNode \*next;

};

//2. List Template

class PList

{

PNode \*head;

public:

PList()

{

head = NULL;

}

void Insert(int c, int e);

void Display();

void Add(PList P,PList Q);

};

//3. Functions

void PList :: Insert(int c, int e)

{

//1. Make a new node

PNode \*t = new PNode;

t->Coeff =c;

t->Exp =e;

t->next =NULL;

//2.

if(head == NULL)

{

head = t;

return;

}

//3.

PNode \* tmp =head;

PNode \*prev =NULL;

while (tmp!=NULL && tmp->Exp > t->Exp)

{

prev = tmp;

tmp =tmp -> next;

}

if( tmp==head) //3a.

{

t->next =head;

}

else if (tmp== NULL) //3b.

{

prev->next =t;

}

else //3c.

{

prev->next =t;

t->next =tmp;

}

}// end of insert

void PList :: Display ()

{

PNode \*tmp =head;

while (tmp)

{

if(tmp->Coeff > 0)

{

cout << "+";

}

cout << tmp -> Coeff <<"x^" <<tmp->Exp;

tmp =tmp->next;

}

}

void PList::Add(PList x, PList y)

{

PNode \*P = x.head;

PNode \*Q = y.head;

int c, e;

while(P!=NULL && Q!=NULL)

{

if(P->Exp==Q->Coeff)

{

c = P->Coeff+Q->Coeff;

if(c!=0)

{

e=P->Exp; //can be Q->Exp also

Insert(c,e);

}

P= P->next;

Q=Q->next;

}//end of if P==Q

else if(P->Exp>Q->Exp)

{

c = P->Coeff;

e = Q->Exp;

Insert(c,e);

P = P->next;

}//end of P>Q

else

{

c = Q->Coeff;

e = Q->Exp;

Insert(c,e);

Q=Q->next;

}//end of P<Q

}//end of P&&Q

while(P!=NULL)

{

c = P->Coeff;

e = P->Exp;

Insert(c,e);

P=P->next;

}

while(Q!=NULL)

{

c= Q->Coeff;

e=Q->Exp;

Insert(c,e);

Q = Q->next;

}

}//end of add

//4. Main

int main()

{

int ch;

PList p1,p2,p3,sum;

int co,ex;

while(1)

{

system("cls");

cout <<"\*\*\*\*Operation on POLnomial\*\*\*\*\n\n";

cout <<"1.Create 1st Polynomial\n";

cout <<"2.Create 2nd Polynomial\n";

cout <<"3.Diplay the 1st Polynomial\n";

cout <<"4.Diplay the 2nd Polynomial\n";

cout <<"5.Add the polynomials\n";

cout <<"6.Exit\n";

cout <<"Enter a choice: ";

cin >>ch;

switch(ch)

{

case 1:

cout << "Enter the coefficent ";

cin >>co;

cout << "Enter the exponenet ";

cin >>ex;

p1.Insert(co,ex);

getch();

break;

case 2:

cout << "Enter the coefficent ";

cin >>co;

cout << "Enter the exponenet ";

cin >>ex;

p2.Insert(co,ex);

getch();

break;

case 3:

cout << "Displayed 1st value ";

p1.Display();

getch();

break;

case 4:

cout << "Displayed 2nd value";

p2.Display();

getch();

break;

case 5:

sum.Add(p1,p2);

cout << "Sum of two polynomials ";

sum.Display();

getch();

break;

case 6:

exit(1);

default:

cout << "Incorrect Choice!";

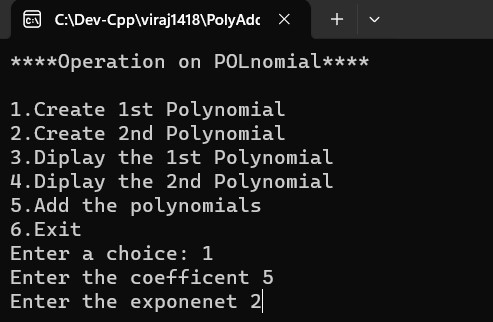
getch();

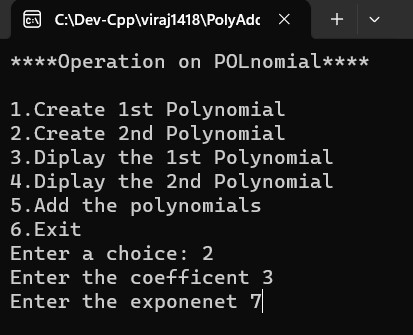
}// end of switch

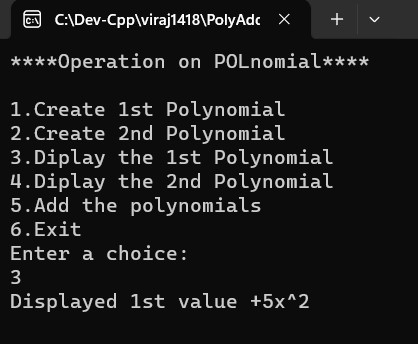
}// end of while

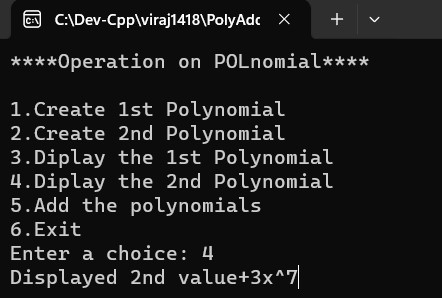
}// end of main

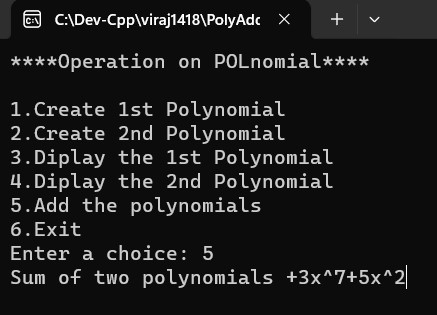
**Output:**











|  |  |
| --- | --- |
| **Program No:** | **24** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 6: Trees – Binary Search Trees** |
| **Objective :** | **Binary Search Trees** |

**Source Code:**

**/\***

Name: Viraj Ghogale

FY-MCA 1418

Unit 6: Trees

Program: Binary Search Trees

**\*/**

**#include<iostream>**

**#include<conio.h>**

**using namespace std;**

**/\*node template\*/**

**class BSTNode**

**{**

**public:**

**int data;**

**BSTNode \*right;**

**BSTNode \*left;**

**};**

**/\* list-based template\*/**

**class BST**

**{**

**BSTNode \*root;**

**int cnt;**

**public:**

**BST()**

**{**

**root = NULL;**

**cnt = 0;**

**}**

**void Insert(int x);**

**void Search(int x);**

**void Display();**

**void Inorder(BSTNode \*p);**

**void Preorder(BSTNode \*p);**

**void Postorder(BSTNode \*p);**

**void Count();**

**void FindMax();**

**void FindMin();**

**};**

**//functions**

**void BST::Insert(int x)**

**{**

**//make a new node t**

**BSTNode \*t = new BSTNode;**

**t->data = x;**

**t->right = NULL;**

**t->left = NULL;**

**//**

**if(root == NULL)**

**{**

**root = t;**

**cnt++;**

**return;**

**}**

**//Traverse**

**BSTNode \*tmp = root;**

**BSTNode \*prev = NULL;**

**while( tmp != NULL)**

**{**

**prev = tmp;**

**if(t->data < tmp->data)**

**{**

**tmp = tmp->left;**

**}**

**else if (t->data > tmp->data)**

**{**

**tmp = tmp->right;**

**}**

**else**

**{**

**cout<<" Duplicate values not aallowed";**

**getch();**

**}**

**}//end of while**

**//Attach t to prev**

**if(t->data < prev->data)**

**{**

**prev->left =t;**

**}**

**else**

**{**

**prev->right =t;**

**}**

**cnt++;**

**}//end of insert**

**void BST :: Display()**

**{**

**cout <<"Inorder traversal: ";**

**Inorder(root);**

**cout << endl;**

**cout <<"Preorder traversal: ";**

**Preorder(root);**

**cout << endl;**

**cout <<"Postorder traversal: ";**

**Postorder(root);**

**cout << endl;**

**}**

**void BST :: Inorder (BSTNode \* p)**

**{**

**if(p!= NULL)**

**{**

**Inorder (p->left);**

**cout<< p->data <<" ";**

**Inorder (p->right);**

**}**

**}**

**void BST :: Preorder (BSTNode \* p)**

**{**

**if(p!= NULL)**

**{**

**cout<< p->data <<" ";**

**Preorder (p->left);**

**Preorder (p->right);**

**}**

**}**

**void BST :: Postorder (BSTNode \* p)**

**{**

**if(p!= NULL)**

**{**

**Postorder (p->left);**

**Postorder (p->right);**

**cout<< p->data <<" ";**

**}**

**}**

**void BST :: Count()**

**{**

**//cnt is declared in the bst class**

**cout << " Number of nodes in BST is : "<<cnt;**

**}//end of count**

**void BST :: Search(int x)**

**{**

**BSTNode \*tmp = root;**

**int flag =0;**

**while( tmp != NULL)**

**{**

**if(x < tmp->data )**

**{**

**tmp= tmp->left;**

**}**

**else if (x > tmp->data )**

**{**

**tmp = tmp->right;**

**}**

**else //Match found**

**{**

**flag=1;**

**break;**

**}**

**}**

**if(flag ==0 )**

**{**

**cout <<"the number is not found : ";**

**}**

**else**

**{**

**cout << x << "number is found! ";**

**}**

**}**

**void BST :: FindMax()**

**{**

**BSTNode \*tmp = root;**

**while(tmp->right != NULL)**

**{**

**tmp = tmp->right;**

**}**

**cout << "Maximum value in the BST: "<<tmp->data;**

**}**

**void BST :: FindMin()**

**{**

**BSTNode \*tmp = root;**

**while(tmp->left!=NULL)**

**{**

**tmp = tmp->left;**

**}**

**cout << " Minimum value in the bst: " <<tmp->data;**

**}**

**//main menu**

**int main()**

**{**

**int ch, num;**

**BST b;**

**while(1)**

**{**

**system("cls");**

**cout<<"\*\*\* Binary Search Tree \*\*\*\n\n";**

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

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

**cout<<"3. Display\n";**

**cout<<"4. Count\n";**

**cout<<"5. FindMax\n";**

**cout<<"6. FindMin\n";**

**cout<<"7. Exit\n";**

**cout<<"Enter an Options\n";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1:**

**cout<<"Insert:\n";**

**cin>>num;**

**b.Insert(num);**

**getch();**

**break;**

**case 2:**

**cout<<"Search";**

**cin>>num;**

**b.Search(num);**

**getch();**

**break;**

**case 3:**

**cout<<"Display\n";**

**b.Display();**

**getch();**

**break;**

**case 4:**

**cout<<"Count";**

**b.Count();**

**getch();**

**break;**

**case 5:**

**b.FindMax();**

**getch();**

**break;**

**case 6:**

**cout<<"FindMin";**

**b.FindMin();**

**getch();**

**break;**

**case 7:**

**exit(1);**

**default:**

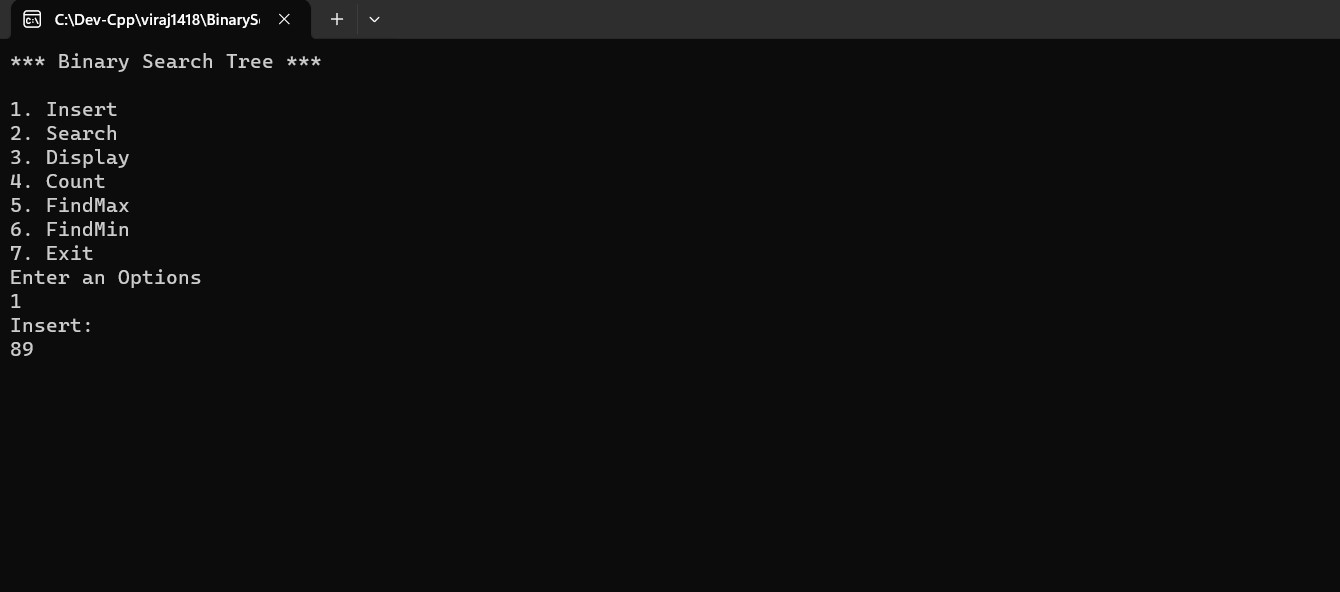
**cout<<"Incorrect choice";**

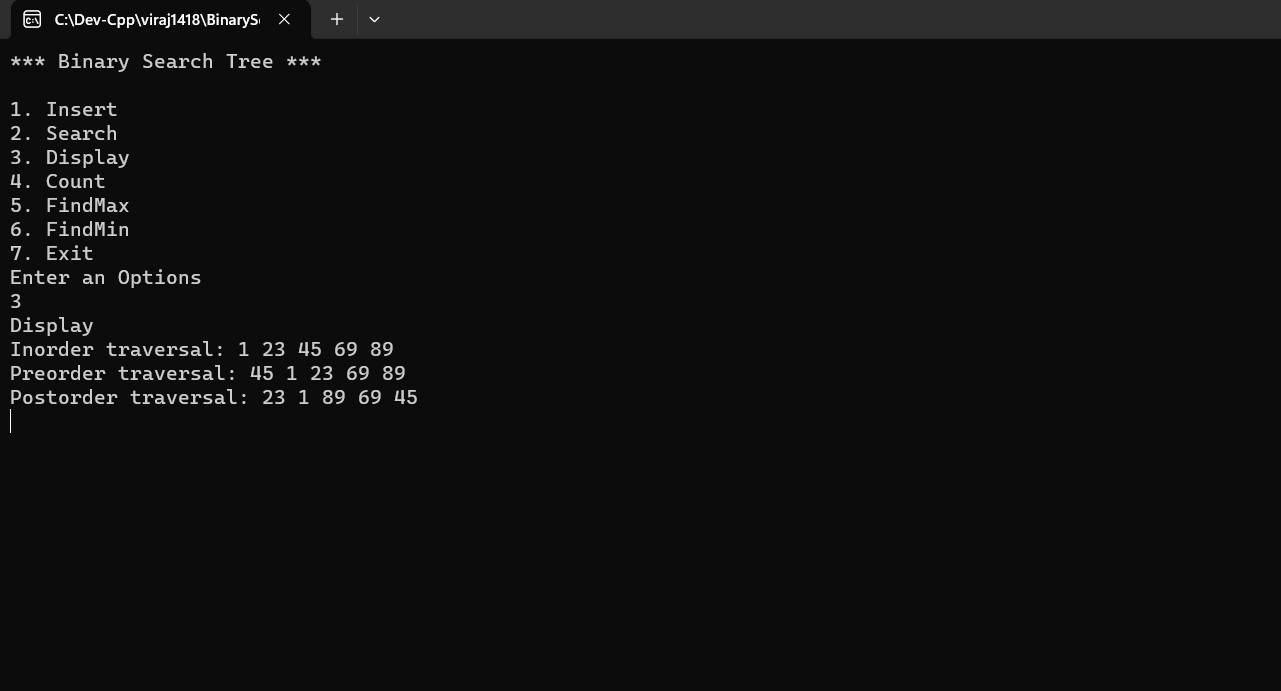
**}**

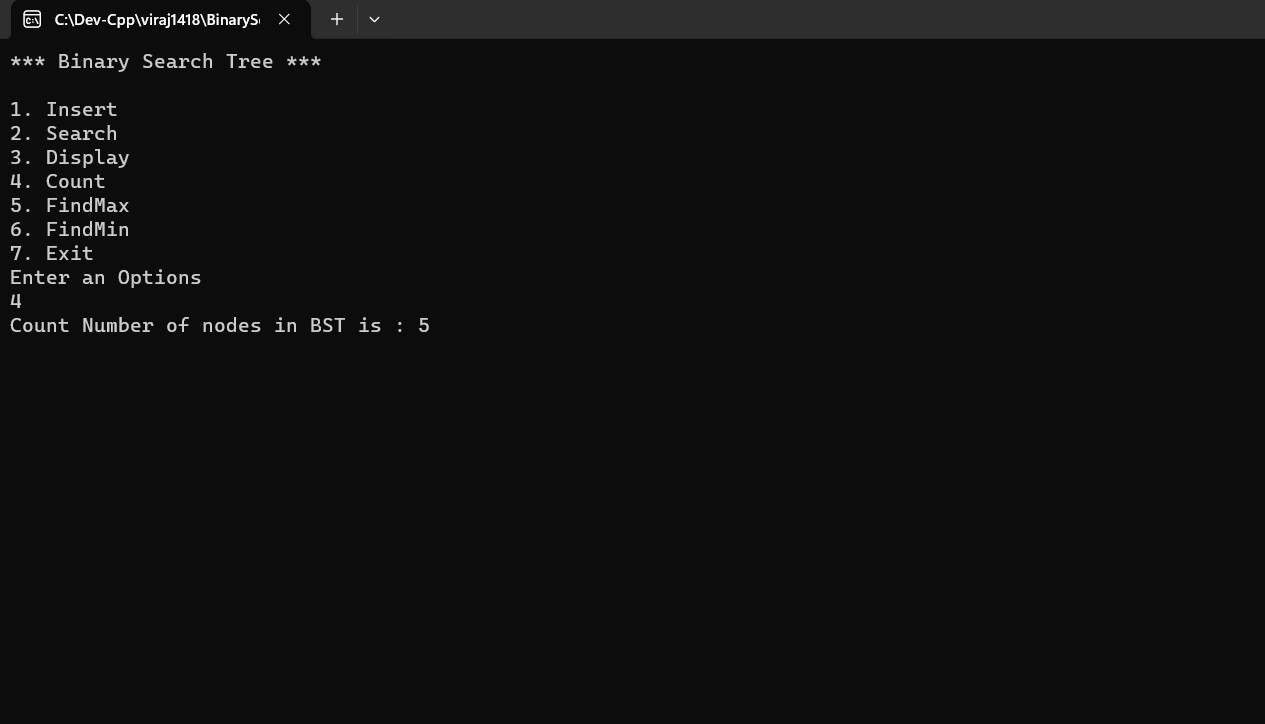
**}**

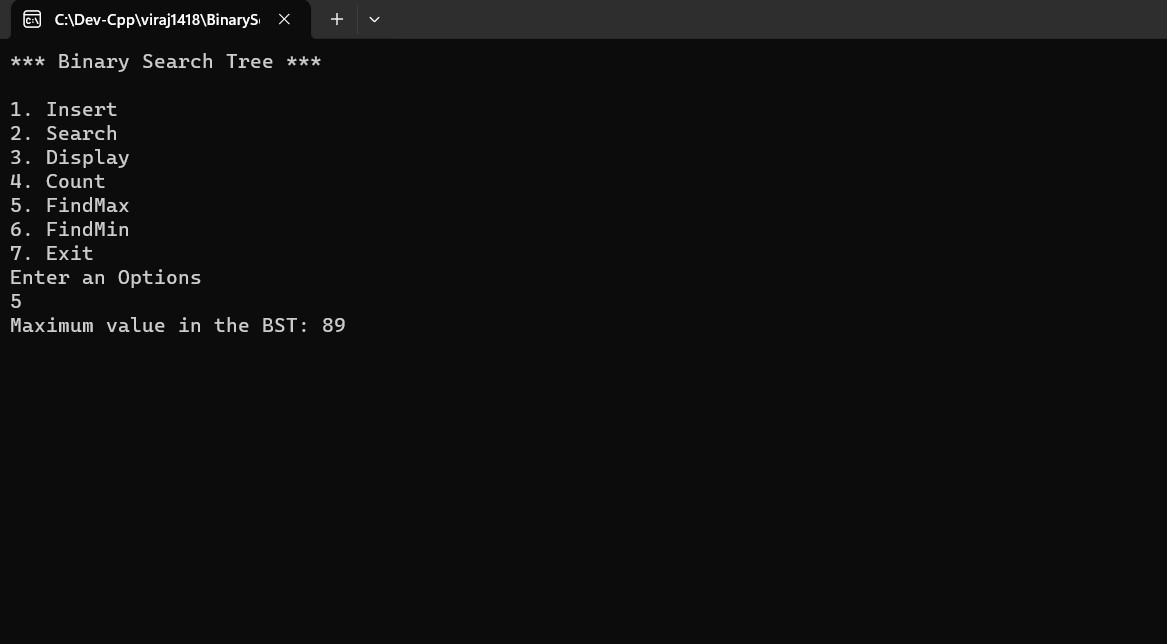
**}//end of main**

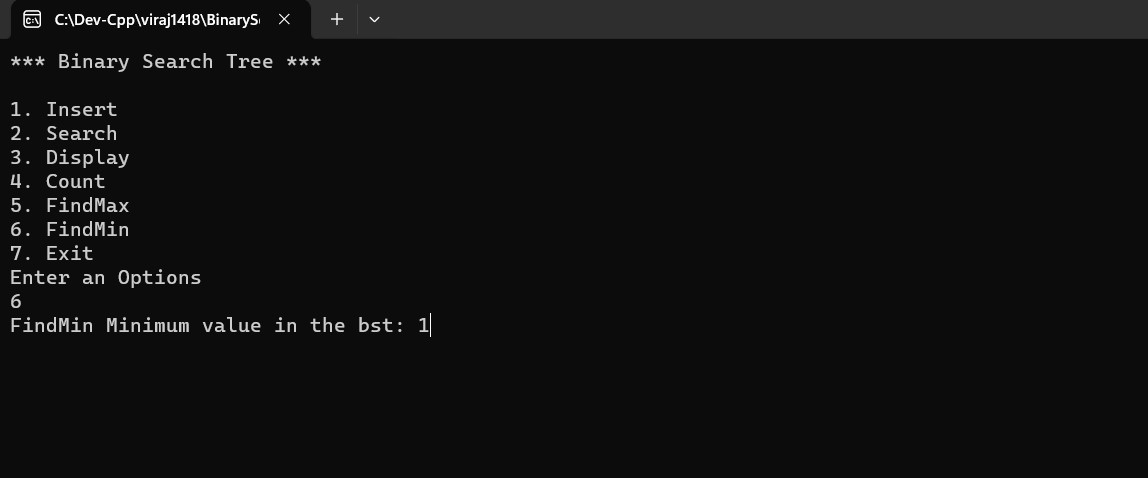
**Output:**

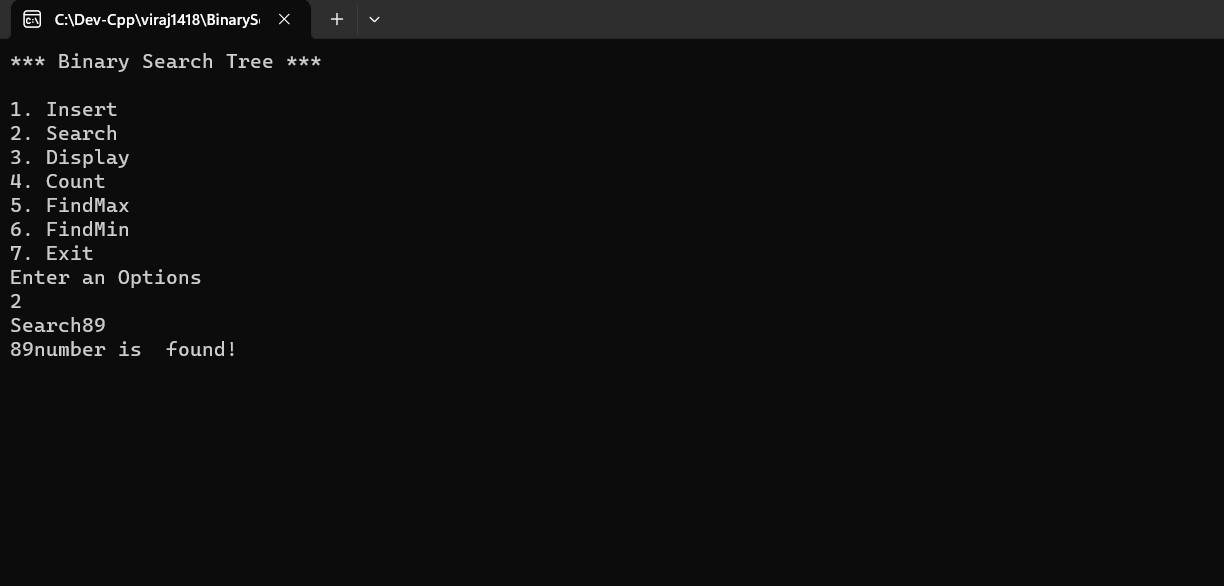












|  |  |
| --- | --- |
| **Program No:** | **25** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 6: Trees – Max-Heaps** |
| **Objective :** | **Max-heaps** |

**Source Code:**

/\*

Name: Viraj Ghogale

FY-MCA 1418

Unit 6: Trees

Program: Max-Heaps

\*/

#include<iostream>

#include<conio.h>

#define MAX 20

using namespace std;

//Node Template

//Array Based Template

class Heap

{

int arr[MAX];

int n;

public:

Heap()

{

int i;

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

{

arr[i] = 0;

}

n=0;

}

void CreateHeap();

void BuildHeap();

void InsertHeap(int x);

void DeleteHeap();

void Display();

void reheapUp(int i);

void reheapDown(int pos);

};

//3.Functions

void Heap :: CreateHeap()

{

int i;

cout << "Enter the number of elements in the heap : ";

cin>> n; //declared in heap

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

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

{

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

cin >> arr[i];

}

BuildHeap();

}

void Heap :: BuildHeap()

{

int i;

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

{

reheapDown(i);

}

}

void Heap :: reheapDown(int pos)

{

int i,val;

val = arr[pos]; //Element to be floated down

while(pos <= (n/2)-1) //Till last non leaf position

{

i = (2\*pos) + 1;

//Find the greater of left and right child

if(arr[i] < arr[i+1])

{

i++; // maintain larger child in i

}

//comppare

if(val > arr[i])

{

break; //nothing to be done

}

arr[pos] = arr[i]; //Shift larger child upwards

//Repeat the same step on the ith index

pos= i;

} //end of while

//copy val to its correct location

arr[pos] = val;

}

void Heap :: Display()

{

if(n == 0)

{

cout << "Heap is empty! ";

return;

}

int i;

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

{

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

}

}

void Heap :: DeleteHeap()

{

//Empty Heap

if(n==0)

{

cout << " Empty Heap ";

return;

}

int tmp = arr[0]; //copy the root element

arr[0] = arr [n-1]; //copy last element to root

arr[n-1]= 0; //Update last element to NIL

n--; //Decrement total number of elements

reheapDown(0); //reheap down the root

cout<<" Element deleted is "<< tmp <<endl;

}

void Heap :: InsertHeap(int x)

{

arr[n] = x; //Insert x at the nth position

n++; //Increment total number of elements

reheapUp(n-1);

}

void Heap :: reheapUp (int i)

{

int val = arr[i]; // copy element to be floated up in thw val

while(i>0 && arr[(i-1/2)] <= val)

{

arr[i] = arr [(i-1)/2];

i = (i-1)/2; //Repeat the same steps on the parent index

}// end of while

arr[i] = val;

}

//4.Menu

int main()

{

int ch,num;

Heap h;

while(1)

{

system("cls");

cout << "\*\*\* Max-Heap \*\*\*\n\n";

cout <<"1. Create Heap\n";

cout <<"2.Insert an element in the heap\n";

cout <<"3.Delete Heap\n";

cout <<"4.Display the heap\n";

cout <<"5.Exit\n";

cout << "Enter your Choice : ";

cin>> ch;

switch(ch)

{

case 1:

h.CreateHeap();

getch();

break;

case 2:

cout <<" Enter the element : ";

cin>> num;

h.InsertHeap(num);

getch();

break;

case 3:

h.DeleteHeap();

getch();

break;

case 4:

h.Display();

getch();

break;

case 5:

exit(1);

default:

cout <<"Incorrect Choice!";

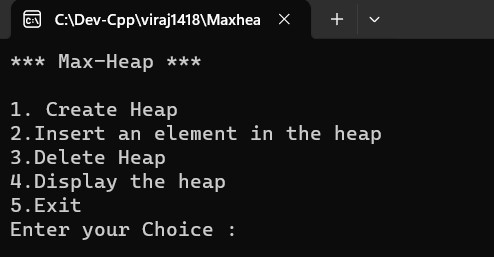
getch();

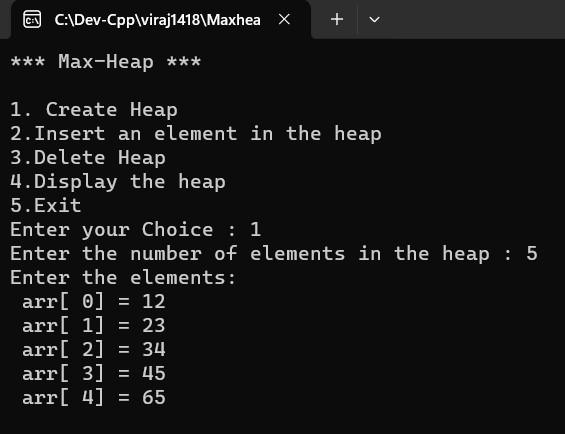
}

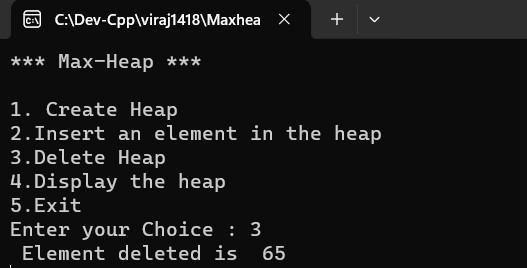
}

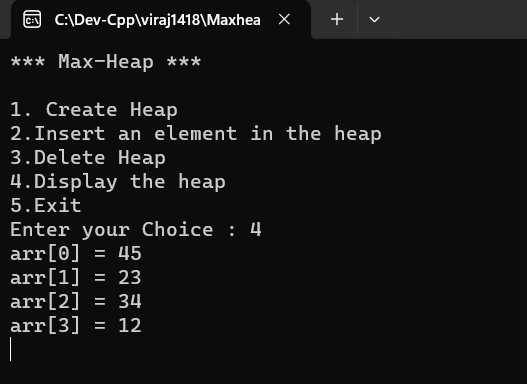
}//end of main

**Output:**









|  |  |
| --- | --- |
| **Program No:** | **26** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 6: Trees – Min-Heaps** |
| **Objective :** | **Min-heaps** |

**Source code:**

**/\***

**Name:Viraj Ghogale**

**RollNo:1418**

**Unit:6**

**Program:MinHeap**

**#include <iostream>**

**#include <conio.h>**

**#define MAX 20**

**using namespace std;**

**// Node Template - NOT REQUIRED**

**// Array Based Template for Min Heap**

**class Heap**

**{**

**int arr[MAX];**

**int n;**

**public:**

**Heap()**

**{**

**int i;**

**for (i = 0; i < MAX; i++)**

**{**

**arr[i] = 0;**

**}**

**n = 0;**

**}**

**void CreateHeap();**

**void BuildHeap();**

**void InsertHeap(int x);**

**void DeleteHeap();**

**void Display();**

**void reheapUp(int i);**

**void reheapDown(int pos);**

**};**

**// Functions**

**void Heap ::CreateHeap()**

**{**

**int i;**

**cout << "Enter the number of elements in the heap: ";**

**cin >> n; // n declared in Heap class**

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

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

**{**

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

**cin >> arr[i];**

**}**

**BuildHeap();**

**}**

**void Heap ::BuildHeap()**

**{**

**int i;**

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

**{**

**reheapDown(i);**

**}**

**}**

**void Heap ::reheapDown(int pos)**

**{**

**int i, val;**

**val = arr[pos]; // Element to be floated down**

**while (pos <= (n / 2) - 1) // Till last leaf position**

**{**

**i = (2 \* pos) + 1; // Left child index**

**// Find the smaller of left and right child**

**if (arr[i] > arr[i + 1] && arr[i + 1] != 0)**

**{**

**i++; // maintain smaller child in i**

**}**

**// Compare val with smaller child**

**if (val < arr[i])**

**{**

**break; // Nothing to be done**

**}**

**arr[pos] = arr[i]; // Shift smaller child upwards**

**// Repeat the same steps on the ith index**

**pos = i;**

**} // end of while**

**// Copy val to its correct location**

**arr[pos] = val;**

**}**

**void Heap ::Display()**

**{**

**if (n == 0)**

**{**

**cout << "Heap is Empty!";**

**return;**

**}**

**int i;**

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

**{**

**cout << "arr[" << i << "] = " << arr[i] << endl;**

**}**

**}**

**void Heap ::DeleteHeap()**

**{**

**// Empty Heap**

**if (n == 0)**

**{**

**cout << "Heap is Empty!";**

**return;**

**}**

**int tmp = arr[0]; // Copy the root element to tmp**

**arr[0] = arr[n - 1]; // Copy last element to root;**

**// arr[n - 1] = 0; // Update last element to NIL**

**n--; // Decrement total number of elements**

**reheapDown(0); // reheapDown on your root;**

**cout << "Element Deleted: " << tmp << endl;**

**}**

**void Heap ::InsertHeap(int x)**

**{**

**arr[n] = x; // Insert x at the nth position**

**n++; // Increment total number of elements**

**reheapUp(n - 1);**

**}**

**void Heap ::reheapUp(int i)**

**{**

**int val = arr[i]; // Copy element to be floated up in val**

**while (i > 0 && arr[(i - 1) / 2] > val)**

**{**

**arr[i] = arr[(i - 1) / 2];**

**i = (i - 1) / 2; // Repeat the same steps on the parent index**

**} // end of while**

**arr[i] = val; // Copy val to its final position**

**}**

**// Menu**

**int main()**

**{**

**int ch, num;**

**Heap h;**

**while (1)**

**{**

**system("cls");**

**cout << "\*\*\* Min - Heap \*\*\*\n\n";**

**cout << "1. Create Heap\n";**

**cout << "2. Insert an element in the Heap\n";**

**cout << "3. Delete Heap\n";**

**cout << "4. Display the Heap\n";**

**cout << "5. Exit\n\n";**

**cout << "Enter your choice: ";**

**cin >> ch;**

**switch (ch)**

**{**

**case 1:**

**h.CreateHeap();**

**getch();**

**break;**

**case 2:**

**cout << "Insert an element in the heap: ";**

**cin >> num;**

**h.InsertHeap(num);**

**getch();**

**break;**

**case 3:**

**h.DeleteHeap();**

**getch();**

**break;**

**case 4:**

**h.Display();**

**getch();**

**break;**

**case 5:**

**exit(1);**

**default:**

**cout << "Incorrect Choice!";**

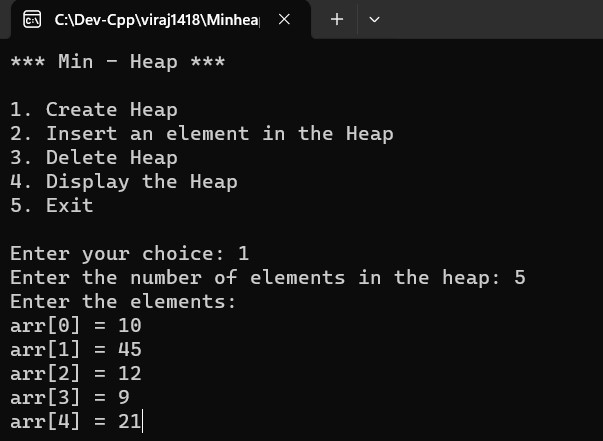
**getch();**

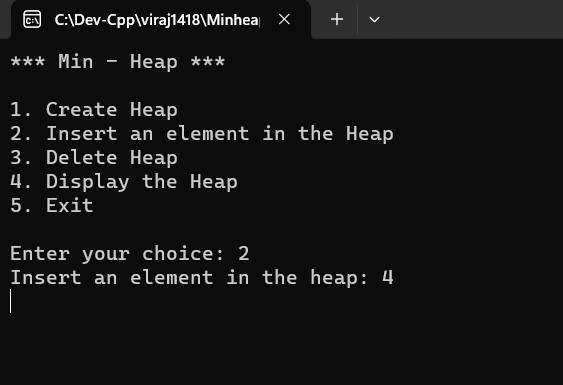
**}**

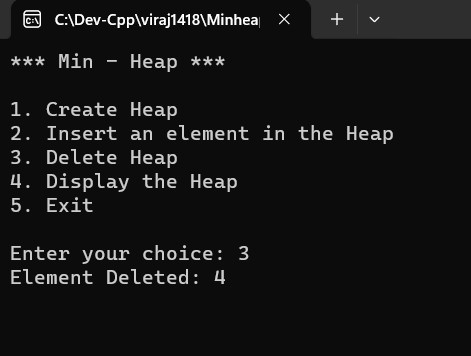
**}**

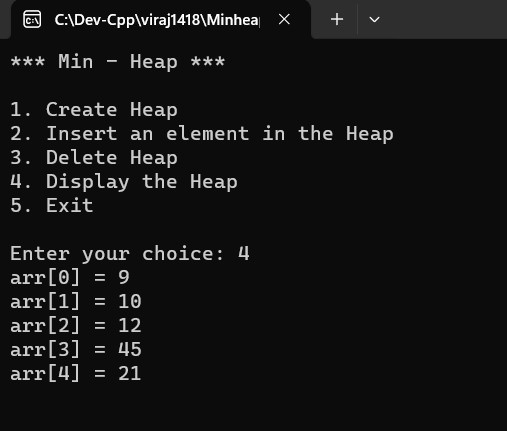
**}**

**Output:**









|  |  |
| --- | --- |
| **Program No:** | **27** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 7: Graphs - Representation of Graphs using Adjacency Matrix** |
| **Objective :** | **Representation of Graphs using Adjacency Matrix** |

**Source Code:**

/\*

Name: Viraj Ghogale

FY-MCA 1418

Unit 7: Graphs

Program: Adjacency Matrix

\*/

#include<iostream>

#include<conio.h>

#define MAX 10

using namespace std;

//1.Node Template-not required

//2.Array Based Template

class Graph

{

int adj[MAX] [MAX];

int n,e;

public:

Graph()

{

int i,j;

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

{

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

{

adj[i][j] =0;

}

}

n=0;

e=0;

}

void CreateGraph();

void Display();

};

//3.Functions

void Graph :: CreateGraph()

{

cout<< "Enter the number of vertrices: ";

cin>>n; //n is declared in the graph class

cout<< "Enter the number of edges : ";

cin >> e; //e is declared in the graph class

int source, dest;

int ne=1;

while(ne<=e)

{

cout <<"Enter edge " << ne << endl;

cout << "Enter source: ";

cin>> source;

cout <<"Enter destination vertex: ";

cin>> dest;

adj[source][dest] =1;

adj[dest][source] =1;

ne++;

}

}

void Graph :: Display()

{

int i , j;

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

{

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

{

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

}

cout<< endl;

}

}

//4.Menu

int main()

{

int ch;

Graph g;

while(1)

{

system("cls");

cout<< "\*\*\* Graph - Adjacency Matrix \*\*\*\n\n";

cout<< "1. Create a graph\n";

cout<< "2. Display the graph\n";

cout<< "3. Exit\n";

cout<< "Enter your choice\n";

cin>>ch;

switch(ch)

{

case 1:

g.CreateGraph();

getch();

break;

case 2:

g.Display();

getch();

break;

case 3:

exit(1);

default:

cout << "Incorrect choice !";

getch();

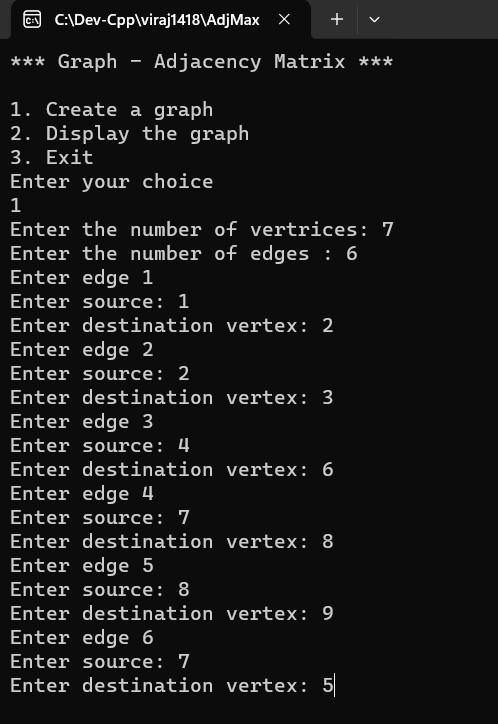
}//end of switch

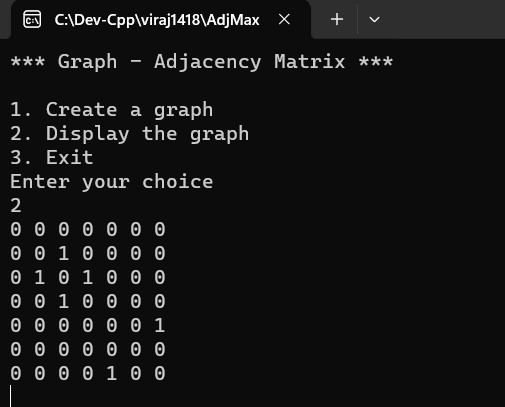
}//end of while

}//end of main

**Output:**







|  |  |
| --- | --- |
| **Program No:** | **28** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 7: Graphs - Perform Breadth-First Traversal on a Graph using Queues** |
| **Objective :** | **Perform Breadth-First Traversal on a Graph using Queues** |

**Source Code:**

/\*

Name: Viraj Ghogale

FY-MCA 1418

Unit 7 - Graphs

Program - Perform Breadth-First Traversal on a Graph using Queues

\*/

#include<iostream>

#include<conio.h>

#define MAX 10

using namespace std;

//Queue Template

class Queue

{

int a[MAX];

int front,rear;

public:

Queue()

{

front = -1;

rear =-1;

}

void Enqueue(int x);

int Dequeue();

int Empty();

};

int Queue :: Empty()

{

if(front == -1)

{

return 1;

}

else

{

return 0;

}

}

void Queue :: Enqueue(int x)

{

rear++;

a[rear] = x;

if(front == -1)

{

front++;

}

}

int Queue :: Dequeue()

{

if(Empty())

{

cout << "Queue Underflow! ";

return -1;

}

int tmp = a[front];

if(front == rear)

{

front = rear =-1;

}

else

{

front++;

}

return tmp;

}

//Graph Template

//1.Node Template-not required

//2.Array Based Template

class Graph

{

int adj[MAX] [MAX];

int visited[MAX];

int n,e;

public:

Graph()

{

int i,j;

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

{

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

{

adj[i][j] =0;

}

}

n=0;

e=0;

}

void CreateGraph();

void Display();

void BFT(int x);

};

//3.Functions

void Graph :: CreateGraph()

{

cout<< "Enter the number of vertrices: ";

cin>>n; //n is declared in the graph class

cout<< "Enter the number of edges : ";

cin >> e; //e is declared in the graph class

int source, dest;

int ne=1;

while(ne<=e)

{

cout <<"Enter edge " << ne << endl;

cout << "Enter source: ";

cin>> source;

cout <<"Enter destination vertex: ";

cin>> dest;

adj[source][dest] =1;

adj[dest][source] =1;

ne++;

}

}

void Graph :: Display()

{

int i , j;

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

{

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

{

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

}

cout<< endl;

}

}

void Graph :: BFT(int x)

{

Queue q;

int i;

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

{

visited[i] =0;

}

//update the starting vertex

visited [x]= 1;

q.Enqueue(x);

cout << "Order of traversal: ";

while(!q.Empty())

{

x= q.Dequeue();

cout << x << " ";

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

{

if(adj [x][i] == 1 && visited[i]==0) //Neighbour & Visisted Status

{

visited[i] = 1;

q.Enqueue(i);

}

}//end of for i

}//end of while

}

//4.Menu

int main()

{

int ch,num;

Graph g;

while(1)

{

system("cls");

cout<< "\*\*\* Graph - Adjacency Matrix \*\*\*\n\n";

cout<< "1. Create a graph\n";

cout<< "2. Display the graph\n";

cout<<" 3.Breadth first traversal\n";

cout<< "4. Exit\n";

cout<< "Enter your choice\n";

cin>>ch;

switch(ch)

{

case 1:

g.CreateGraph();

getch();

break;

case 2:

g.Display();

getch();

break;

case 3:

cout <<"Enter starting vertex: ";

cin>>num;

g.BFT(num);

getch();

break;

case 4:

exit(1);

default:

cout << "Incorrect choice !";

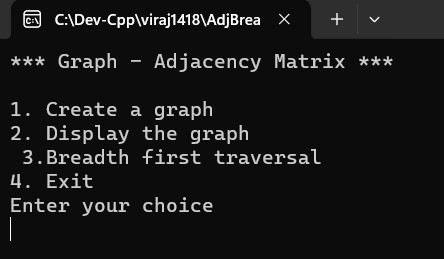
getch();

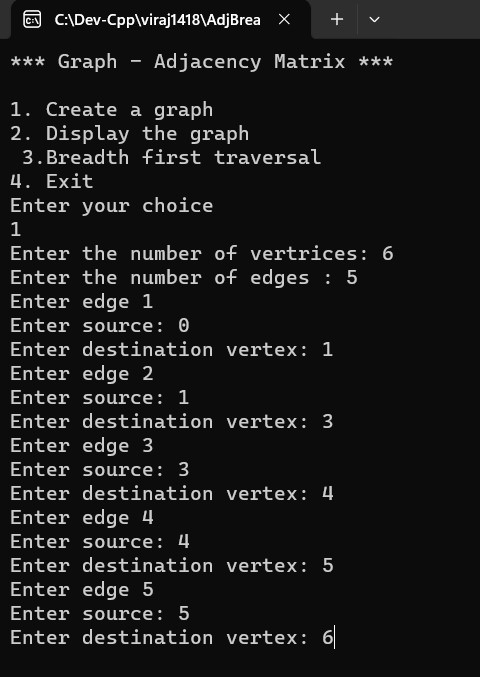
}//end of switch

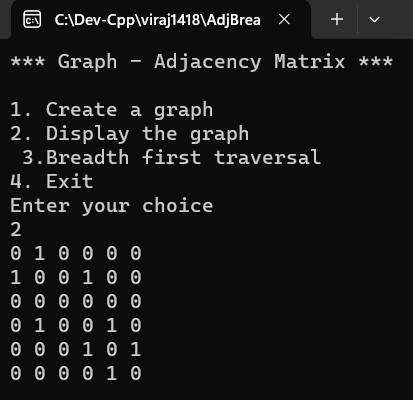
}//end of while

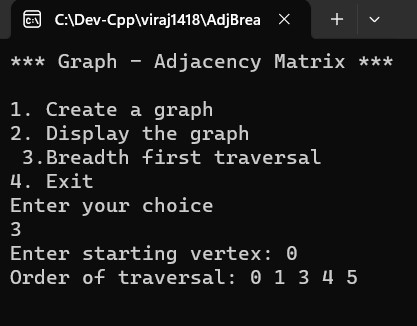
}//end of main

**Output:**









|  |  |
| --- | --- |
| **Program No:** | **29** |
| **Roll No :** | **1418** |
| **Title of Program :** | **Unit 6: Trees – Find MST using Kruskal’s Algorithm** |
| **Objective :** | **Find MST using Kruskal’s Algorithm** |

**Source Code:**

/\*

Name: Viraj Ghogale

FY-MCA 1418

Unit 7: Graphs

Program: MST - Kruskal's Algortihm

\*/

#include<iostream>

#include<conio.h>

#define MAX 10

#define INF 999

using namespace std;

//1.Node Template-not required

//2.Array Based Template

class Graph

{

int adj[MAX] [MAX];

int n,e;

int parent[MAX];

public:

Graph()

{

int i,j;

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

{

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

{

adj[i][j] =INF;

}

}

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

{

parent[i] = -1;

}

n=0;

e=0;

}

void CreateGraph();

void Display();

void Kruskal();

int Find(int i);

int Union(int i, int j);

};

//3.Functions

void Graph :: CreateGraph()

{

cout<< "Enter the number of vertrices: ";

cin>>n; //n is declared in the graph class

cout<< "Enter the number of edges : ";

cin >> e; //e is declared in the graph class

int source, dest, weight;

int ne=1;

while(ne<=e)

{

cout <<"Enter edge " << ne << endl;

cout << "Enter source: ";

cin>> source;

cout <<"Enter destination vertex: ";

cin>> dest;

cout <<"Enter weight: ";

cin>> weight;

adj[source][dest] =weight;

adj[dest][source] =weight;

ne++;

}

}

void Graph :: Display()

{

int i , j;

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

{

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

{

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

}

cout<< endl;

}

}

void Graph :: Kruskal()

{

int minimum,i,j;

int ne=1;

int a,b,u,v;

int mincost = 0;

while(ne < n)

{

//Find the minimum edge

for(i=0, minimum =999; i<n ; i++)

{

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

{

if(adj[i][j] < minimum)

{

minimum = adj[i][j];

a = u = i;

b = v = j;

}//end for j

}

}// end for i

u = Find(u);

v = Find(v);

if(Union(u,v))

{

cout << " Edge : " << ne++ << ": (" <<a << "," << b << ") =" <<adj[a][b]<<endl;

mincost +=minimum;

}

adj[a][b] = adj[b][a] = INF; //To ensure edge is not selected again

}//end of while

cout <<endl<<"MST Total Cost: " << mincost;

}

int Graph :: Find(int i)

{

while(parent[i]!= -1)

{

i = parent[i];

}

return i;

}

int Graph :: Union(int i, int j)

{

if(i!=j)

{

parent[j] = i;

return 1;

}

}

//4.Menu

int main()

{

int ch;

Graph g;

while(1)

{

system("cls");

cout<< "\*\*\* Graph - Minimum Spaning Tree \*\*\*\n\n";

cout<< "1. Create a graph\n";

cout<< "2. Display the graph\n";

cout<< "3. MST- Kruskal's Algorithim'\n";

cout<< "4. Exit\n";

cout<< "Enter your choice\n";

cin>>ch;

switch(ch)

{

case 1:

g.CreateGraph();

getch();

break;

case 2:

g.Display();

getch();

break;

case 3:

g.Kruskal();

getch();

break;

case 4:

exit(1);

default:

cout << "Incorrect choice !";

getch();

}//end of switch

}//end of while

}//end of main

**Output:**

