**INTERNATIONAL INSTITUTE OF**

**PROFESSIONAL STUDIES (IIPS), INDORE**

DS Lab PROGRAMMING

ASSIGNMENT

SUBMITTED BY – Khushbu Nemade

ROLL NO. – IT-2K20-26

SUBMITTED TO :–

Nidhi Sethi ma’am

ENROLL NO. – DE2002034

M.tech(IT) IIlrd Sem

**INDEX**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1. | [**Program to illustrate use of object and class**](#object_class) | 3 |
| 2. | [**Program to illustrate constructors and its types default, parameterized and copy constructor**](#constructor) | 5 |
| 3. | [**Program to print array values which user has entered.**](#array_p3) | 9 |
| 4. | [**Program to print array values using pointers**](#array_p4) | 11 |
| 5. | [**Program to implement dynamic array.**](#array_p5) | 13 |
| 6. | [**Program to illustrate Matrix addition.**](#matrix_p6) | 15 |
| 7. | [**Program for bubble sort.**](#array_p7) | 19 |
| 8. | [**Program to insert new elements in an already filled array.**](#array_p8) | 22 |
| 9. | [**Program to delete any element from an array**](#array_p9) | 25 |
| 10. | [**Program for insertion sort.**](#sort_p10) | 28 |
| 11. | [**Program for selection sort**](#sort_p11) | 31 |
| 12. | [**Program for linear search**](#search_p12) | 34 |
| 13. | [**Program for binary search.**](#search_p13) | 37 |
| 14. | [**Program to merge two sorted arrays**](#sort_p14) | 41 |
| 15. | [**Program to merge two unsorted array than to create a new sorted array.**](#merge_p15) | 46 |
| 16. | [**Program to implement merge sort**](#merge_p16) | 50 |
| 17. | [**Program to implement quick sort**](#sor_p17) | 55 |
| 18. | [**Program to implement stack using arrays.**](#stack_p18) | 59 |
| 19. | [**Program to create linked list.**](#list_p19) | 64 |
| 20. | [**Program to insert an element in the linked list(first, middle and last)**](#list_p20) | 65 |
| 21. | [**Program to find out length of linked list.**](#list_p21) | 69 |
| 22. | [**Program to search an element in a linked list.**](#list_p22) | 60 |
| 23. | [**Program to delete an element in the linked list(first, middle and last)**](#list_p23) | 63 |
| 24. | [**Program to create sorted list.**](#list_p24) | 64 |
| 25. | [**Program to delete duplicates from sorted list**](#list_p25) | 65 |
| 26. | [**Program to delete duplicates from unsorted list**](#list_p26) | 66 |
|  |  |  |
| 27. | [**Program to reverse a linked list.**](#list_p28) | 67 |
| 28. | [**Program to create a list**](#list_p29) | 89 |
| 29. | [**Program to insert a node at beginning , middle , last**](#list_p30) | 92 |
| 30. | [**Program to reverse doubly linked list**](#list_p31) | 94 |
| 31. | [**Program to delete a node at beginning, middle, last**](#list_p32) | 95 |
| 32. | [**Program to count nodes of doubly linked list**](#list_p33) | 98 |
| 33. | [**Program to check list is Palindrome or not**](#list_p34) | 98 |
| 34. | [**Program to find largest node**](#list_p35) | 99 |
| 35. | [**Program to search a node in list**](#list_p36) | 100 |
| 36. | [**Program to search a node in list backward**](#list_p37) | 100 |
| 38. | [**Program to remove duplicate nodes**](#list_p38) | 102 |
| 39. | [**Program to sort doubly linked list**](#list_p39) | 103 |
| 40. | [**Program to display list**](#list_p40) | 103 |
| 41. | [**Program to convert infix to postfix expression**](#list_p41) | 121 |
| 42. | [**Program to evaluate postfix expression**](#list_p42) | 125 |
| 43. | [**Program to implement queue using array**](#list_p43) | 128 |
| 44. | [**Program to implement circular queue**](#list_p44) | 131 |
| 45. | [**Program to implement priority queue**](#list_p45) | 137 |
| 46. | [**Program to implement Dqueue**](#list_p46) | 141 |
| 47. | [**Program to implement Binary tree**](#list_p47) | 148 |
| 48. | [**Program for inorder traversal**](#list_p48) | 150 |
| 49. | [**Program for preorder traversal**](#list_p49) | 151 |
| 50. | [**Program for postorder traversal**](#list_p50) | 152 |
| 51. | [**Program to insert a node in binary search tree**](#list_p51) | 153 |
| 52. | [**Program to delete a node in binary search tree**](#list_p52) | 158 |
| 53. | [**Program to implement max heap**](#list_p53) | 164 |
| 54. | [**Program to implement min heap**](#list_p54) | 166 |
| 55. | [**Program to insert a node in max heap**](#list_p55) | 169 |
| 56. | [**Program to delete a node in max heap**](#list_p56) | 171 |
| 57. | [**Program to insert a node in min heap**](#list_p57) | 173 |
| 58. | [**Program to delete a node in min heap**](#list_p58) | 176 |
|  |  |  |

**// Program to illustrate use of objects and class**

#include <iostream>

using namespace std;

class Employee {

public:

int id;//data member

string name;//data member

float salary;

void insert(int i, string n, float s)

{

id = i;

name = n;

salary = s;

}

void display()

{

cout<<id<<" "<<name<<" "<<salary<<endl;

}

};

int main(void) {

Employee e1;

Employee e2;

e1.insert(27, "Khushbu",120000);

e2.insert(28, "Manisha", 800000);

e1.display();

e2.display();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

27 Khushbu 120000

28 Manisha 800000

**//program to demonstrate the use of default constructor**

#include<iostream>

using namespace std;

class cons

{

int i;

public:

cons()

{

cout<<"Default constructor"<<endl;

i=30;

}

cons(int n)

{

cout<<"\nParametrised constructor"<<endl;

i=n;

}

cons(cons &c)

{

cout<<"\nCopy constructor"<<endl;

i=c.i;

}

void display()

{

cout<<i;

}

};

int main()

{

cons c1;

c1.display();

cons c2(40);

c2.display();

cons c3=c1;

c3.display();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ programe2.cpp -o programe2 } ; if ($?) { .\programe2 }

Default constructor

30

Parametrised constructor

40

Copy constructor

30

**//Program to print array values which user has entered.**

#include<iostream>

using namespace std;

class array

{

int arr[5],n;

public:

void takedata();

void print();

};

void array::print()

{

cout<<" Entered Array : ";

for(int i=0;i<5;i++)

{

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

}

}

void array::takedata()

{

cout<<"Enter array elements:";

for(int i=0;i<5;i++)

{

cin>>arr[i];

}

}

int main()

{

array a1;

a1.takedata();

a1.print();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter array elements:2

4

6

3

5

Entered Array : 2 4 6 3 5

**//Program to print array values using pointers**

#include<iostream>

using namespace std;

class array

{

int \*p,a[20],size;

public:

array()

{

p=a;

cout<<"\nEnter size of array:";

cin>>size;

cout<<"\nEnter elements of array:";

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

{

cin>>p[i];

}

}

void display()

{

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

{

cout<<p[i]<<" ";

}

}

};

int main()

{

array a;

cout<<"\nEntered array elements are:";

a.display();

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter size of array:4

Enter elements of array:2

8

5

7

Entered array elements are:2 8 5 7

**// Program to implement dynamic array.**

#include<iostream>

using namespace std;

class array

{

int\*p;

int size;

public:

array()

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

cin>>size;

cout<<"Enter array elements: ";

p=new int[size];

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

{

cin>>p[i];

}

}

void print()

{

cout<<"\nEntered array elements are:";

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

{

cout<<p[i]<<" ";

}

}

~array()

{

cout<<"\nDestructor called!"<<endl;

delete p;

}

};

int main()

{

array a1;

a1.print();

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter size of array : 4

Enter array elements: 9

3

7

5

Entered array elements are:9 3 7 5

Destructor called!

**//Program to illustrate Matrix addition.**

#include<iostream>

using namespace std;

class array

{

int n,r,c,add[10][10],a[10][10],b[10][10];

public:

array()

{

cout<<"Enter number of rows :";

cin>>r;

cout<<"\nEnter number of columns:";

cin>>c;

}

void input();

void print();

void addition();

};

void array::input()

{

//for matrix A

cout<<"Enter elements of array for matrix A:- "<<endl;

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

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

cin>>a[i][j];

//for matric B

cout<<"Enter elements of array for matrix B:- "<<endl;

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

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

cin>>b[i][j];

}

void array::addition()

{

cout<<"Adiition of matrix (A+B):- "<<endl;

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

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

add[i][j]= a[i][j]+b[i][j];

}

void array::print()

{

//Print addition

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

{

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

{

cout<<add[i][j]<<"\t";}

cout<<endl;

}

}

int main()

{

array a1;

a1.input();

a1.addition();

a1.print();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter number of rows :2

Enter number of columns:2

Enter elements of array for matrix A:-

9

3

8

5

Enter elements of array for matrix B:-

8

6

7

5

Adiition of matrix (A+B):-

17 9

15 10

**// BUBBLE SORT**

#include<iostream>

using namespace std;

void display(int a[], int n)

{

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

{

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

}

cout << endl;

}

void bsort(int a[], int n)

{

int temp;

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

{

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

{

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

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

}

int main()

{

int size;

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

cin >> size;

int arr[size];

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

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

{

cin >> arr[i];

}

cout << "Before sorting : " << endl;

display(arr, size);

//sorting fuction

bsort(arr, size);

cout << "After sorting : " << endl;

display(arr, size);

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ bubblesort.cpp -o bubblesort } ; if ($?) { .\bubblesort

Enter the size of array : 5

Enter the elements of array :

1

4

5

6

2

Before sorting :

1 4 5 6 2

After sorting :

1 4 5 6 2

**// Program to insert new elements in an already filled array.**

#include <iostream>

using namespace std;

class array

{

public:

int arr[50], i, element, position, n;

void Array()

{

cout << "Enter the Size for Array: " << endl;

cin >> n;

cout << "Enter " << n << " Array Elements: " << endl;

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

cin >> arr[i];

}

void insert()

{

cout << "\nEnter Element to Insert: " << endl;

cin >> element;

cout << "At What Position ? : " << endl;

cin >> position;

for (i = n; i >= position; i--)

arr[i] = arr[i - 1];

arr[i] = element;

n++;

}

void display()

{

cout << "\nThe New Array is:\n";

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

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

cout << endl;

}

};

int main()

{

array arr;

arr.Array();

arr.insert();

arr.display();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ insert\_element.cpp -o insert\_element } ; if ($?) { .\insert\_element }Enter the Size for Array:

5

Enter 5 Array Elements:

1

5

4

2

4

Enter Element to Insert:

9

At What Position ? :

3

The New Array is:

1. 5 9 4 2 4

**// Program to delete any element from an array**

#include <iostream>

using namespace std;

class del\_element

{

public:

int arr[100], tot, i, elem, j, found = 0;

del\_element()

{

cout << "Enter the Size: "<<endl;

cin >> tot;

cout << "Enter " << tot << " Array Elements: "<<endl;

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

cin >> arr[i];

}

void dalete()

{

cout << "\nEnter Element to Delete: \n";

cin >> elem;

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

{

if (arr[i] == elem)

{

for (j = i; j < (tot - 1); j++)

arr[j] = arr[j + 1];

found = 1;

i--;

tot--;

}

}

}

void dispaly()

{

if (found == 0)

cout << "\nElement doesn't found in the Array!";

else

{

cout << "\nElement Deleted Successfully!";

cout << "\n\nThe New Array is:\n";

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

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

}

cout << endl;

}

};

int main()

{

del\_element D;

D.dalete();

D.dispaly();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter array element

2

8

4

6

5

2 8 4 6 5

Enter position to delete

3

2 8 4 5

**// Program for insertion sort.**

#include <iostream>

using namespace std;

class sort

{

int i, j, n, temp, a[30];

public:

void input()

{

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

cin >> n;

cout << "\nEnter the elements\n";

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

{

cin >> a[i];

}

}

void isort()

{

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

{

temp = a[i];

j = i - 1;

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

{

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

j = j - 1;

}

a[j + 1] = temp;

}

}

void display()

{

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

{

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

}

}

};

int main()

{

sort s;

int n;

s.input();

cout << "Entered list is :\n";

s.display();

s.isort();

cout << "\n\nSorted list is \n";

s.display();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ insertionsort.cpp -o insertionsort } ; if ($?) { .\insertionsort }

Enter the number of elements:5

Enter the elements

1

5

24

6

8

Entered list is :

1 5 24 6 8

Sorted list is

1 5 6 8 24

**// Program for selection sort**

#include <iostream>

using namespace std;

class sort

{

int i, j, n, temp, a[30];

public:

void input()

{

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

cin >> n;

cout << "\nEnter the elements\n";

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

{

cin >> a[i];

}

}

void isort()

{

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

{

temp = a[i];

j = i - 1;

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

{

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

j = j - 1;

}

a[j + 1] = temp;

}

}

void display()

{

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

{

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

}

}

};

int main()

{

sort s;

int n;

s.input();

cout << "Entered list is :\n";

s.display();

s.isort();

cout << "\n\nSorted list is \n";

s.display();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ selectionsort.cpp -o selectionsort } ; if ($?) { .\selectionsort }

Enter the number of elements:5

Enter the elements

1

2

5

2

8

Entered list is :

1 2 5 2 8

Sorted list is

1 2 2 5 8

**// Program for linear search**

#include<iostream>

using namespace std;

int linearsearch(int arr[],int n,int key)

{

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

{

if(arr[i]==key)

{

return i;

}

}

}

int main()

{

int n;

cout<<"Enter number of elements"<<endl;

cin>>n;

int arr[n];

cout<<"Enter array elements:";

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

{

cin>>arr[i];

}

int key;

cout<<"Enter key element to search:"<<endl;

cin>>key;

cout<<"The position of key element in array is:";

cout<<linearsearch(arr,n,key)<<endl;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter number of elements

3

Enter array elements:9

4

2

Enter key element to search:

4

The position of key element in array is:1

**// Program for binary search.**

#include<iostream>

using namespace std;

int binarysearch(int arr[],int n,int key)

{

int s = 0;

int e = n;

while(s<=e)

{

int mid = (s+e)/2;

if(arr[mid]==key)

{

return mid;

}

else if(arr[mid]>key)

{

e=mid-1;

}

else{

s = mid+1;

}

}

return -1;

}

int main()

{

int n;

cout<<"Enter number of elements"<<endl;

cin>>n;

int arr[n];

cout<<"Enter array elements:";

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

{

cin>>arr[i];

}

int key;

cout<<"Enter key element to search:"<<endl;

cin>>key;

cout<<"The position of key element in array is:";

cout<<binarysearch(arr,n,key)<<endl;

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter number of elements

3

Enter array elements:9

3

7

Enter key element to search:

3

The position of key element in array is:1

**// Program to merge two sorted arrays**

#include <iostream>

using namespace std;

class array

{

int i, j, k;

public:

void display(int ar[], int);

void merge(int ar1[], int ar2[], int, int);

};

void array ::display(int ar[], int n)

{

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

{

cout << ar[i] << " ";

}

cout << endl;

}

void array ::merge(int a1[], int a2[], int n1, int n2)

{

int a[n1 + n2];

i = j = k = 0;

while (i < n1 && j < n2)

{

if (a1[i] < a2[j])

{

a[k] = a1[i];

i++;

k++;

}

else

{

a[k] = a2[j];

k++;

j++;

}

}

while (i < n1)

{

a[k] = a1[i];

k++;

i++;

}

while (j < n2)

{

a[k] = a2[j];

k++;

j++;

}

cout << "\nAfter merging the two sorted arrays, we get :" << endl;

for (i = 0; i < n1 + n2; i++)

{

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

}

}

int main()

{

array arr;

int s1, s2;

cout << "\nEnter the size of first sorted array : ";

cin >> s1;

int arr1[s1];

cout << "\nEnter the elements of first array in sorted order : ";

for (int i = 0; i < s1; i++)

{

cin >> arr1[i];

}

cout << "\nEnter the size of second sorted array : ";

cin >> s2;

int arr2[s2];

cout << "\nEnter the elements of second array in sorted order : ";

for (int i = 0; i < s2; i++)

{

cin >> arr2[i];

}

cout << "\nThe elements of first sorted array are : " << endl;

arr.display(arr1, s1);

cout << "\nThe elements of second sorted array are : " << endl;

arr.display(arr2, s2);

cout << "" << endl;

//Calling the merging function

arr.merge(arr1, arr2, s1, s2);

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ mergesort.cpp -o mergesort } ; if ($?) { .\mergesort }

Enter the size of first sorted array : 5

Enter the elements of first array in sorted order : 4

5

6

7

8

Enter the size of second sorted array : 4

Enter the elements of second array in sorted order : 1

2

3

4

The elements of first sorted array are :

4 5 6 7 8

The elements of second sorted array are :

1 2 3 4

After merging the two sorted arrays, we get :

1 2 3 4 4 5 6 7 8

**// Program to merge two unsorted array than to create a new sorted array.**

#include <iostream>

using namespace std;

void sort(int a[], int size)

{

for (int i = 0; i < size - 1; i++)

{

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

{

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

{

int temp = a[j];

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

a[j + 1] = temp;

}

}

}

}

void merge(int a[], int b[], int size1, int size2)

{

int newarr[size1 + size2];

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

while (i < size1 && j < size2)

{

if (a[i] < b[j])

{

newarr[k] = a[i];

k++;

i++;

}

else

{

newarr[k] = b[j];

k++;

j++;

}

}

while (i < size1)

{

newarr[k] = a[i];

i++;

k++;

}

while (j < size2)

{

newarr[k] = b[j];

j++;

k++;

}

for (int i = 0; i < (size1 + size2); i++)

{

cout << newarr[i] << " ";

}

}

int main()

{

int a[100];

int b[100], size1, size2;

cout << "enter the size of first array " << endl;

cin >> size1;

cout << "enter the elements of first array" << endl;

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

{

cin >> a[i];

}

cout << "enter the size of second array " << endl;

cin >> size2;

cout << "enter the elements of second array" << endl;

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

{

cin >> b[i];

}

sort(a, size1);

sort(b, size2);

merge(a, b, size1, size2);

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ mergeunsortedarr.cpp -o mergeunsortedarr } ; if ($?) { .\mergeunsortedarr }enter the size of first array

5

enter the elements of first array

1

5

4

8

2

enter the size of second array

2

enter the elements of second array

5

2

1 2 2 4 5 5 8

**// Program to implement merge sort**

#include <iostream>

using namespace std;

class msort

{

int arr[10];

public:

int n;

int low, high;

void getdata();

void mergesort(int low, int high);

void combine(int low, int mid, int high);

void display();

};

void msort::getdata()

{

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

cin >> n;

cout << "\nEnter array elements:" << endl;

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

cin >> arr[i];

}

void msort::mergesort(int low, int high)

{

int mid;

if (low < high)

{

mid = (low + high) / 2;

mergesort(low, mid);

mergesort(mid + 1, high);

combine(low, mid, high);

}

}

void msort::combine(int low, int mid, int high)

{

int i, j, k;

int temp[10];

k = low;

i = low;

j = mid + 1;

while (i <= mid && j <= high)

{

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

{

temp[k] = arr[i];

i++;

k++;

}

else

{

temp[k] = arr[j];

j++;

k++;

}

}

while (i <= mid)

{

temp[k] = arr[i];

i++;

k++;

}

while (j <= high)

{

temp[k] = arr[j];

j++;

k++;

}

for (k = low; k <= high; k++)

arr[k] = temp[k];

}

void msort::display()

{

cout << "\n\nSorted array is ";

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

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

}

int main()

{

msort obj;

cout << "\n----- Merge sort: ------ \n";

obj.getdata();

obj.low = 0;

obj.high = obj.n - 1;

obj.mergesort(obj.low, obj.high);

obj.display();

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ mergesort.cpp -o mergesort } ; if ($?) { .\mergesort }

----- Merge sort: ------

Enter the lenth of array : 5

Enter array elements:

1

4

5

2

6

Sorted array is : 1 2 4 5 6

**// Program to implement quick sort**

#include <iostream>

using namespace std;

class sort

{

public:

void quick\_sort(int a[], int l, int u)

{

int j;

if (l < u)

{

j = partition(a, l, u);

quick\_sort(a, l, j - 1);

quick\_sort(a, j + 1, u);

}

}

int partition(int a[], int l, int u)

{

int v, i, j, temp;

v = a[l];

i = l;

j = u + 1;

do

{

do

i++;

while (a[i] < v && i <= u);

do

j--;

while (v < a[j]);

if (i < j)

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

} while (i < j);

a[l] = a[j];

a[j] = v;

return (j);

}

};

int main()

{

sort s;

int a[50], n, i;

cout << "\nHow many elements you want in an array : ";

cin >> n;

cout << "\nEnter array elements : ";

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

cin >> a[i];

s.quick\_sort(a, 0, n - 1);

s.partition(a, 0, n - 1);

cout << "\nArray after sorting : ";

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

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

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ quicksort.cpp -o quicksort } ; if ($?) { .\quicksort }

How many elements you want in an array : 5

Enter array elements : 1

5

4

2

6

Array after sorting : 1 2 4 5 6

**// Program to implement stack using arrays.**

#include <iostream>

using namespace std;

class stack

{

int stack[100], n = 100, top = -1;

public:

void push(int val)

{

if (top >= n - 1)

cout << "Stack Overflow " << endl;

else

{

top++;

stack[top] = val;

}

}

void pop()

{

if (top <= -1)

cout << "Stack Underflow " << endl;

else

{

cout << "The popped element is : " << stack[top] << endl;

top--;

}

}

void display()

{

if (top >= 0)

{

cout << "Stack elements are : ";

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

cout << stack[i] << " ";

cout << endl;

}

else

cout << "Stack is empty ";

}

};

int main()

{

stack s;

int ch, val;

cout << "1) Push in stack" << endl;

cout << "2) Pop from stack" << endl;

cout << "3) Display stack" << endl;

cout << "4) Exit" << endl;

do

{

cout << "Enter choice : " << endl;

cin >> ch;

switch (ch)

{

case 1:

{

cout << "Enter value to be pushed : " << endl;

cin >> val;

s.push(val);

break;

}

case 2:

{

s.pop();

break;

}

case 3:

{

s.display();

break;

}

case 4:

{

cout << "Exit" << endl;

break;

}

default:

{

cout << "Invalid Choice" << endl;

}

}

} while (ch != 4);

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ arraystack.cpp -o arraystack } ; if ($?) { .\arraystack }

1) Push in stack

2) Pop from stack

3) Display stack

4) Exit

Enter choice :

1

Enter value to be pushed :

5

Enter choice :

1

Enter value to be pushed :

5

Enter choice :

1

Enter value to be pushed :

2

Enter choice :

3

Stack elements are : 2 5 5

Enter choice :

2

The popped element is : 2

Enter choice :

2

The popped element is : 5

Enter choice :

3

Stack elements are : 5

Enter choice :

4

Exit

**// Program to create linked list.**

#include <iostream>

using namespace std;

struct node

{

int data;

node \*next;

node \*current = NULL;

};

class linkedlist

{

node \*head;

public:

linkedlist()

{

head = NULL;

}

void insert(int);

void prepend(int);

void append(int);

void addafter(int, int);

void del(int);

void sorting();

void deleteduplicate();

void deleteunsortdulicate();

void reverse();

void display();

void count();

bool search(int);

~linkedlist()

{

node \*temp = head;

while (head != NULL)

{

temp = head;

head = head->next;

delete temp;

}

}

};

**// Program to insert an element in the linked list**

void linkedlist::insert(int n)

{

int length;

cout << "\n Enter the length of node/data which u want to fill : " << endl;

cin >> length;

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

{

cout << "\n Enter the value # " << i + 1 << " : ";

cin >> n;

node \*temp = new node, \*t;

temp->data = n;

temp->next = NULL;

if (head == NULL)

{

head = temp;

t = temp;

}

else

{

t->next = temp;

t = t->next;

}

}

}

void linkedlist::prepend(int n)

{

node \*temp;

temp = new node;

temp->data = n;

temp->next = head;

head = temp;

}

void linkedlist::append(int n)

{

node \*temp, \*t;

temp = new node;

temp->data = n;

temp->next = NULL;

if (head == NULL)

{

head = temp;

}

else

{

t = head;

while (t->next != NULL)

{

t = t->next;

}

t->next = temp;

}

}

void linkedlist::addafter(int loc, int n)

{

node \*temp, \*t;

temp = new node;

temp->data = n;

t = head;

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

{

t = t->next;

temp->next = t->next;

}

t->next = temp;

}

**// Program to delete an element in the linked list**

void linkedlist::del(int n)

{

node \*t, \*old;

t = head;

while (t != NULL)

{

if (t->data == n)

{

if (t == head)

{

head = t->next;

}

else

{

old->next = t->next;

delete t;

return;

}

}

else

{

old = t;

t = t->next;

}

}

cout << "\nelement ot found";

}

**// Program to create sorted list.**

void linkedlist::sorting()

{

int temp;

node \*ptr, \*t;

ptr = head;

while (ptr->next != NULL)

{

t = ptr->next;

while (t != NULL)

{

if (ptr->data > t->data)

{

temp = ptr->data;

ptr->data = t->data;

t->data = temp;

}

t = t->next;

}

ptr = ptr->next;

}

}

**// Program to delete duplicates from sorted list .**

void linkedlist::deleteduplicate()

{

node \*curr = head;

node \*next\_n;

if (curr == NULL)

{

return;

}

while (curr->next != NULL)

{

if (curr->data == curr->next->data)

{

next\_n = curr->next->next;

free(curr->next);

curr->next = next\_n;

}

else

{

curr = curr->next;

}

}

}

**// Program to delete duplicates from unsorted list .**

void linkedlist::deleteunsortdulicate()

{

node \*t1, \*t2, \*temp;

t1 = head;

while (t1 != NULL && t1->next != NULL)

{

t2 = t1;

while (t2->next != NULL)

{

if (t1->data == t2->next->data)

{

temp = t2->next;

t2->next = t2->next->next;

delete (temp);

}

else

{

t2 = t2->next;

}

}

t1 = t1->next;

}

}

**// Program to reverse a linked linkedlist**

void linkedlist::reverse()

{

node \*prev, \*current, \*nextnode;

if (head == NULL)

{

cout << "\nList is Empty : ";

return;

}

if (head->next == NULL)

{

cout << "only one node is there : ";

return;

}

current = head;

prev = NULL;

while (current != NULL)

{

nextnode = current->next;

current->next = prev;

prev = current;

current = nextnode;

}

head = prev;

}

void linkedlist::display()

{

if (head == NULL)

{

cout << "\n list is empty";

}

node \*t;

t = head;

while (t != NULL)

{

cout << " " << t->data;

t = t->next;

}

}

**// Program to find out length of linked list.**

void linkedlist::count()

{

if (head == NULL)

{

cout << "\nlist is empty";

}

else

{

node \*t;

int counter = 0;

t = head;

while (t != NULL)

{

counter++;

t = t->next;

}

cout << "\n Total no of node is :" << counter << endl

<< endl;

}

}

**// Program to search an element in a linked list.**

bool linkedlist::search(int n)

{

node \*tmp;

tmp = head;

while (tmp != NULL)

{

if (tmp->data == n)

{

return true;

}

tmp = tmp->next;

}

return false;

}

int main()

{

linkedlist ls;

int n;

ls.insert(n);

ls.display();

ls.count();

int option;

do

{

cout << "\nSelect the options." << endl;

cout << "1. appendNode()" << endl;

cout << "2. prependNode()" << endl;

cout << "3. add after()" << endl;

cout << "4. delete()" << endl;

cout << "5. search()" << endl;

cout << "6. sorting()" << endl;

cout << "7. remove duplicates in sorted list" << endl;

cout << "8. remove duplicates in unsorted list" << endl;

cout << "9. revrse()" << endl;

cout << "10. distructor called()" << endl;

cout << "11. Clear Screen" << endl;

cout << "12. End loop" << endl

<< endl;

cin >> option;

switch (option)

{

case 1:

cout << "\n Enter the value which u want to append : ";

cin >> n;

ls.append(n);

ls.display();

ls.count();

break;

case 2:

cout << "\n Enter the value which u want to prepend : ";

cin >> n;

ls.prepend(n);

ls.display();

ls.count();

break;

case 3:

int loc;

cout << "\n Enter the location and the value where u wnd to add a data :";

cout << "location : ";

cin >> loc;

cout << "value : ";

cin >> n;

ls.addafter(loc, n);

ls.display();

ls.count();

case 4:

cout << "\n Enter the value which u want to delete : ";

cin >> n;

ls.del(n);

ls.display();

ls.count();

break;

case 5:

cout << "enter the value which u wnat to find : ";

cin >> n;

if (ls.search(n))

{

cout << "\nEntered value is found ";

}

else

{

cout << "\nEnter value is not found ";

}

break;

case 6:

cout << "\n After Srting : ";

ls.sorting();

ls.display();

ls.count();

break;

case 7:

cout << "\nRemove all duplicates in sorted linked list : ";

ls.deleteduplicate();

ls.display();

ls.count();

break;

case 8:

cout << "\nRemove all Duplicates in Unsorted linked list : ";

ls.deleteunsortdulicate();

ls.display();

ls.count();

break;

case 9:

cout << "\nReversed list is : ";

ls.reverse();

ls.display();

ls.count();

break;

case 10:

cout << "\n Destructor Calleed : ";

ls.~linkedlist();

ls.display();

ls.count();

break;

case 11:

system("cls");

break;

case 12:

break;

default:

cout << "Enter Proper Option number " << endl;

}

} while (option != 12);

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ linkedlist.cpp -o linkedlist } ; if ($?) { .\linkedlist }

Enter the length of node/data which u want to fill :

5

Enter the value # 1 : 4

Enter the value # 2 : 5

Enter the value # 3 : 7

Enter the value # 4 : 8

Enter the value # 5 : 1

4 5 7 8 1

Total no of node is :5

Select options.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

1

Enter the value which u want to append : 9

4 5 7 8 1 9

Total no of node is :6

What operation do you want to perform? Select Option number. Enter 0 to exit.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

2

Enter the value which u want to prepend : 8

8 4 5 7 8 1 9

Total no of node is :7

What operation do you want to perform? Select Option number. Enter 0 to exit.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

3

Enter the location and the value where u wnd to add a data :location : 4

value : 5

8 4 5 7 8 5 1 9

Total no of node is :8

Enter the value which u want to delete : 4

8 5 7 8 5 1 9

Total no of node is :7

What operation do you want to perform? Select Option number. Enter 0 to exit.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

5

enter the value which u wnat to find : 5

Entered value is found

Select option.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

6

After Srting : 1 5 5 7 8 8 9

Total no of node is :7

Select options.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

7

Remove all duplicates in sorted linked list : 1 5 7 8 9

Total no of node is :5

Select options.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

8

Remove all Duplicates in Unsorted linked list : 1 5 7 8 9

Total no of node is :5

Select options.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

9

Reversed list is : 9 8 7 5 1

Total no of node is :5

What operation do you want to perform? Select Option number. Enter 0 to exit.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

10

Destructor Called :

list is empty

list is empty

Select options.

1. appendNode()

2. prependNode()

3. add after()

4. delete()

5. search()

6. sorting()

7. remove duplicates in sorted list

8. remove duplicates in unsorted list

9. revrse()

10. distructor called()

11. Clear Screen

12. End loop

12

Thank You

**// Program to implement stack using linked list.**

#include <iostream>

using namespace std;

struct Node

{

int data;

Node \*next;

};

class stack

{

Node \*top;

public:

stack()

{

top = NULL;

}

void push(int n)

{

Node \*newnode = new Node;

newnode->data = n;

newnode->next = top;

top = newnode;

}

void pop()

{

if (top == NULL)

cout << "Stack Underflow" << endl;

else

{

cout << "The popped element is : " << top->data << endl;

top = top->next;

}

}

void display()

{

Node \*ptr;

if (top == NULL)

{

cout << "stack is empty";

}

else

{

ptr = top;

cout << "Stack elements are : ";

while (ptr != NULL)

{

cout << ptr->data << " ";

ptr = ptr->next;

}

}

cout << endl;

}

};

int main()

{

stack s;

int ch, n;

cout << "1) Push in stack" << endl;

cout << "2) Pop from stack" << endl;

cout << "3) Display stack" << endl;

cout << "4) Exit" << endl;

do

{

cout << "Enter choice: " << endl;

cin >> ch;

switch (ch)

{

case 1:

{

cout << "Enter value to be pushed:" << endl;

cin >> n;

s.push(n);

break;

}

case 2:

{

s.pop();

break;

}

case 3:

{

s.display();

break;

}

case 4:

{

cout << "Exit" << endl;

break;

}

default:

{

cout << "Invalid Choice" << endl;

}

}

} while (ch != 4);

return 0;

}

**OUTPUT :-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ stacklinkedist.cpp -o stacklinkedist } ; if ($?) { .\stacklinkedist }

1) Push in stack

2) Pop from stack

3) Display stack

4) Exit

Enter choice:

1

Enter value to be pushed:

2

Enter choice:

1

Enter value to be pushed:

4

Enter choice:

1

Enter value to be pushed:

5

Enter choice:

3

Stack elements are : 5 4 2

Enter choice:

2

The popped element is : 5

Enter choice:

2

The popped element is : 4

Enter choice:

3

Stack elements are : 2

Enter choice:

4

Exit

**//Program to Create Doubly linked list**

#include<iostream>

using namespace std;

struct node

{

int data;

node \*prev ,\*next;

};

class dlinkedlist{

node \*head ,\*end;

public:

dlinkedlist(){

head=end=NULL;

}

void insert(int);

void addAtBeg(int);

void addAtlast(int);

void addafter(int,int);

void delAtBegning();

void delATLast();

void del(int);

bool search(int);

void search\_backward(int);

void remove\_duplicate();

bool palindrome();

void sort();

int lagrest\_node();

void reverse();

void display();

void traverse\_backward();

int count();

};

void dlinkedlist::insert(int n)

{

int length;

cout << "\n Enter the length of list : " << endl;

cin >> length;

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

{

cout << "\n Enter the value # " << i + 1 << " : ";

cin >> n;

node \*t, \*temp;

temp = new node;

temp->data = n;

temp->next = NULL;

if (head == NULL)

{

temp->prev = NULL;

head = temp;

}

else

{

t = head;

while (t->next != NULL)

t = t->next;

t->next = temp;

temp->prev = t;

}

}

}

**//Program to insert a node at beginning , middle , last**

void dlinkedlist::addAtBeg(int n){

node \*temp=new node;

temp->data=n;

temp->prev=NULL;

temp->next=head;

if(head!=NULL){

head->prev=temp;

}

head=temp;

}

void dlinkedlist::addAtlast(int n){

node \*temp,\*t;

temp=new node;

temp->prev=NULL;

temp->data=n;

temp->next=NULL;

t=head;

while(t->next!=NULL){

t=t->next;

}

t->next=temp;

temp->prev=t;

}

void dlinkedlist::addafter(int loc,int n){

int c=count();

if(loc<0 || loc>c){

cout<<"Invalid Loction"<<endl;

}

else{

if(loc==0){

addAtBeg(n);

}

else if(loc==c){

addAtlast(n);

}

else{

node \*temp,\*t;

temp=new node;

temp->data=n;

t=head;

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

t=t->next;

}

t->next->prev=temp;

temp->next=t->next;

temp->prev=t;

t->next=temp;

}

}

}

**//Program to reverse doubly linked list**

void dlinkedlist::reverse()

{

node \*t1, \*t2;

t1 = head;

t2 = t1->next;

t1->next = NULL;

t1->prev = t2;

while (t2 != NULL)

{

t2->prev = t2->next;

t2->next = t1;

t1 = t2;

t2 = t2->prev;

}

head = t1;

cout<<" Reversed List"<<endl;

}

**//Program to delete a node at beginning, middle, last**

void dlinkedlist::delAtBegning(){

node\* tmp;

if(head==NULL){

cout<<"Under FLow"<<endl;

}

else if(head->next==NULL){

head=NULL;

free(head);

cout<<"Node Deleted!!!"<<endl;

}

else{

tmp=head;

head=head->next;

head->prev=NULL;

free(tmp);

cout<<"Node Deleted!!!"<<endl;

}

}

void dlinkedlist::del(int n){

node \*temp=head;

while(temp!=NULL){

if(temp->data==n){

if(temp==head){

if(head->next==NULL){

head=end=NULL;

}

else{

head=head->next;

head->prev=NULL;

}

}

else{

if(temp->next==NULL){

temp->prev->next=NULL;

end=temp->prev;

}

else{

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

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

}

}

delete temp;

return;

}

temp=temp->next;

}

cout<<"\nElemenet not found : "<<endl;

}

void dlinkedlist::delATLast(){

node \*tmp,\*t;

if(head==NULL){

cout<<"Under Flow"<<endl;

}

else if(head->next==NULL){

free(head);

head=NULL;

cout<<"Node Deleted!!!"<<endl;

}

else{

tmp=head;

while(tmp->next!=NULL){

t=tmp;

tmp=tmp->next;

}

free(tmp);

t->next=NULL;

cout<<"Node Deleted!!!"<<endl;

}

}

**//Program to count nodes of doubly linked list**

int dlinkedlist::count(){

node \*temp=head;

int c=0;

while(temp!=NULL){

c++;

temp=temp->next;

}

return c;

}

**//Program to check list is Palindrome or not**

bool dlinkedlist::palindrome(){

node \*left=head;

if(left==NULL){

return true;

}

node \*right=left;

while(right->next!=NULL){

right=right->next;

}

while(left!=right && right!=left->prev)

{

if(left->data!=right->data){

cout<<"\n The List Is Not A Palimdorme "<<endl;

return false;

}

left=left->next;

right=right->prev;

}

cout<<"\n The List Is Palindrome "<<endl;

return true;

}

**//Program to find largest node in list**

int dlinkedlist::lagrest\_node(){

node \*max,\*tmp;

tmp=max=head;

while (tmp!=NULL)

{

if(tmp->data > max->data){

max=tmp;

}

tmp=tmp->next;

}

cout<<endl<<max->data<<endl;

return max->data;

}

**//Program to search a node in list**

bool dlinkedlist::search(int n){

node \*tmp;

tmp=head;

while(tmp!=NULL){

if(tmp->data==n){

cout<<"\nElement fount at position : "<<tmp->data<<endl;

return true;

}

tmp=tmp->next;

}

cout<<"\n Element not found !! ";

return false;

}

**//Program to search a node in list backward**

void dlinkedlist::search\_backward(int n){

node\* tmp;

int pos=0;

tmp=head;

while(tmp->next!=NULL){

tmp=tmp->next;

}

while(tmp!=NULL){

pos++;

if(tmp->data==n){

cout<<"Found at position : "<<tmp->data<<endl;

return ;

}

if(tmp->prev!=NULL){

tmp=tmp->prev;

}

else{

break;

}

cout<<"Element Not Found !!"<<endl;

return ;

}

}

**//Program to remove duplicate nodes**

void dlinkedlist::remove\_duplicate(){

node\* tmp,\*t,\*i;

if(head==NULL){

cout<<"List is Empty : "<<endl;

return;

}

else{

for(tmp=head;tmp!=NULL;tmp=tmp->next){

for(i=tmp->next;i!=NULL;i=i->next){

if(tmp->data==i->data){

t=i;

i->prev->next=i->next;

if(i->next!=NULL){

i->next->prev=i->prev;

t=NULL;

}

}

}

}

}

}

**//Program to sort doubly linked list**

void dlinkedlist::sort(){

node \*tmp=NULL,\*t=NULL;

int p;

if(head==NULL){

cout<<"List is Empty : ";

return;

}

else{

for(tmp=head;tmp->next!=NULL;tmp=tmp->next){

for(t=tmp->next;t!=NULL;t=t->next){

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

p=tmp->data;

tmp->data=t->data;

t->data=p;

}

}

}

}

}

**//Program to Display doubly linked list**

void dlinkedlist::display(){

if(head==NULL)

cout<<"List is empty";

node \*t;

t=head;

cout<<endl;

while(t!=NULL)

{

cout<<" "<<t->data;

t=t->next;

}

}

int main(){

dlinkedlist dl;

int n,choice,loc;

dl.insert(n);

dl.display();

node \*head;

while (1)

{

cout<<"\nOperations on Doubly linked list"<<endl;

cout<<"1.Add at begning"<<endl;

cout<<"2.Add at last"<<endl;

cout<<"3.Add after position"<<endl;

cout<<"4.Delete At beginning"<<endl;

cout<<"5.Delete"<<endl;

cout<<"6.Delete At Last"<<endl;

cout<<"7.Remove Duplicates"<<endl;

cout<<"8.Palindrome()"<<endl;

cout<<"9.Sort the list"<<endl;

cout<<"10.Largest Node"<<endl;

cout<<"11.Reverse"<<endl;

cout<<"12.Search()"<<endl;

cout<<"13.Search From Backward direction"<<endl;

cout<<"14.Display"<<endl;

cout<<"15.Count"<<endl;

cout<<"16.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch ( choice )

{

case 1:

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

cin>>n;

dl.addAtBeg(n);

dl.display();

cout<<endl;

break;

case 2:

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

cin>>n;

dl.addAtlast(n);

dl.display();

cout<<endl;

break;

case 3:

cout<<"\nInsert Element after postion: \n";

cin>>loc;

cout<<"\nEnter the element: \n";

cin>>n;

dl.addafter(loc,n);

dl.display();

cout<<endl;

break;

case 4:

dl.delAtBegning();

dl.display();

break;

case 5:

if (head == NULL)

{

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

break;

}

cout<<"Enter the element to delete : \n"<<endl;

cin>>n;

dl.del(n);

dl.display();

cout<<endl;

break;

case 6:

dl.delATLast();

dl.display();

break;

case 7:

dl.remove\_duplicate();

cout<<"\nAfter Removing Duplicates"<<endl;

dl.display();

break;

case 8:

dl.palindrome();

dl.display();

break;

case 9:

cout<<"\nSorted list is : "<<endl;

dl.sort();

dl.display();

break;

case 10:

cout<<"\nLargest Node is : "<<endl;

dl.lagrest\_node();

dl.display();

break;

case 11:

if (head == NULL)

{

cout<<"List is empty!!\n"<<endl;

break;

}

dl.reverse();

dl.display();

cout<<endl;

break;

case 12:

cout<<"\nEnter element which you want to search : \n";

cin>>n;

dl.search(n);

break;

case 13:

cout<<"\nEnter element which you want to search : \n";

cin>>n;

dl.search\_backward(n);

dl.display();

break;

case 14:

dl.display();

cout<<endl;

break;

case 15:

cout<<"\nNumber of elements are: \n"<<dl.count()<<endl;

break;

case 16:

exit(1);

default:

cout<<"Wrong choice\n"<<endl;

}

}

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ alldlinkedlist.cpp -o alldlinkedlist } ; if ($?) { .\alldlinkedlist }

Enter the length of list :

4

Enter the value # 1 : 2

Enter the value # 2 : 3

Enter the value # 3 : 4

Enter the value # 4 : 5

2 3 4 5

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 1

Enter the element:

1

1 2 3 4 5

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 2

Enter the element:

6

1 2 3 4 5 6

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 3

Insert Element after postion:

2

Enter the element:

7

1 2 7 3 4 5 6

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 4

Node Deleted!!!

2 7 3 4 5 6

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 6

Node Deleted!!!

2 7 3 4 5

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 7

After Removing Duplicates

2 7 3 4 5

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 8

The List Is Not A Palimdorme

2 7 3 4 5

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 9

Sorted list is :

2 3 4 5 7

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 10

Largest Node is :

7

2 3 4 5 7

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 11

Reversed List

7 5 4 3 2

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 12

Enter element which you want to search :

5

Element fount at position : 5

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 13

Enter element which you want to search :

2

Found at position : 2

7 5 4 3 2

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

Enter your choice : 14

7 5 4 3 2

Operations on Doubly linked list

1.Add at begning

2.Add at last

3.Add after position

4.Delete At beginning

5.Delete

6.Delete At Last

7.Remove Duplicates

8.Palindrome()

9.Sort the list

10.Largest Node

11.Reverse

12.Search()

13.Search From Backward direction

14.Display

16.Count

17.Quit

**//Program to convert infix to postfix expression**

#include<iostream>

#include<stack>

using namespace std;

int prec(char c) {

if(c == '^')

return 3;

else if(c == '/' || c=='\*')

return 2;

else if(c == '+' || c == '-')

return 1;

else

return -1;

}

void infixToPostfix(string s) {

stack<char> st;

string result;

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

char c = s[i];

if((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))

result += c;

else if(c == '(')

st.push('(');

else if(c == ')') {

while(st.top() != '(')

{

result += st.top();

st.pop();

}

st.pop();

}

else {

while(!st.empty() && prec(s[i]) <= prec(st.top())) {

result += st.top();

st.pop();

}

st.push(c);

}

}

while(!st.empty()) {

result += st.top();

st.pop();

}

cout << result << endl;

}

int main() {

string exp = "a-b/(c\*d+e)";

infixToPostfix(exp);

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

abcd\*e+/-

**//Program to evaluate postfix expression**

#include<stack>

#include<math.h>

#include<iostream>

using namespace std;

int postfixEvaluation(string s)

{

stack<int>st;

for(int i=0;i<s.length();i++)

{

if(s[i]>='0'&& s[i]<='9')

{

st.push(s[i]-'0');

}

else

{

int op2 = st.top();

st.pop();

int op1 = st.top();

st.pop();

switch (s[i])

{

case '+':

st.push(op1+op2);

break;

case '-':

st.push(op1-op2);

break;

case '\*':

st.push(op1\*op2);

break;

case '/':

st.push(op1/op2);

break;

case '^':

st.push(pow(op1,op2));

break;

}

}

}

return st.top();

}

int main()

{

cout<<postfixEvaluation("46+2/5\*7+")<<endl;

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

32

**//Program to implement Queue using array**

#include<iostream>

using namespace std;

const int MAX = 5;

class queue{

int arr[MAX],front,rear;

public:

queue()

{

front = rear = -1;

}

void addq(int);

int delq();

};

void queue::addq(int item)

{

if(rear==MAX-1)

cout<<"queue is full";

else

{

rear++;

arr[rear]=item;

if(front==-1)

front = 0;

}

}

int queue::delq()

{

int data;

if(front==-1)

{

cout<<"Queue is empty";

//retun -1;

}

data=arr[front];

arr[front]=-1;

if(front==rear)

front=rear=-1;

else

front++;

return data;

}

int main()

{

queue q1;

q1.addq(2);

q1.addq(5);

int x=q1.delq();

q1.addq(8);

x=q1.delq();

cout<<" "<<x;

x=q1.delq();

cout<<" "<<x;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

5 8

**//Program to implement circular queue**

#include<iostream>

#define SIZE 100

using namespace std;

class node

{

public:

node()

{

next = NULL;

}

int data;

node \*next;

}\*front=NULL,\*rear=NULL,\*n,\*temp,\*temp1;

class cqueue

{

public:

void insertion();

void deletion();

void display();

};

int main()

{

cqueue c1;

int ch;

do

{

cout<<"operations::";

cout<<"\n1. Insert\n2. Delete\n3. Display\n4. Exit\n\nEnter Your Choice: ";

cin>>ch;

switch(ch)

{

case 1:

c1.insertion();

c1.display();

break;

case 2:

c1.deletion();

break;

case 3:

c1.display();

break;

case 4:

break;

default:

cout<<"\nWrong Choice!!! ";

}

}while(ch!=4);

return 0;

}

void cqueue::insertion()

{

n=new node[sizeof(node)];

cout<<"\nEnter the Element: ";

cin>>n->data;

if(front==NULL)

{

front=n;

}

else

{

rear->next=n;

}

rear=n;

rear->next=front;

}

void cqueue::deletion()

{

int x;

temp=front;

if(front==NULL)

{

cout<<"\nCircular Queue Empty!!!";

}

else

{

if(front==rear)

{

x=front->data;

delete(temp);

front=NULL;

rear=NULL;

}

else

{

x=temp->data;

front=front->next;

rear->next=front;

delete(temp);

}

cout<<"\nElement "<<x<<" is Deleted";

display();

}

}

void cqueue::display()

{

temp=front;

temp1=NULL;

if(front==NULL)

{

cout<<"\n\nCircular Queue Empty!!!";

}

else

{

cout<<"\n\nCircular Queue Elements are:\n\n";

while(temp!=temp1)

{

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

temp=temp->next;

temp1=front;

}

}

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ circular-queueLL.cpp -o circular-queueLL } ; if ($?) { .\circular-queueLL }

operations::

1. Insert

2. Delete

3. Display

4. Exit

Enter Your Choice: 1

Enter the Element: 3

Circular Queue Elements are:3

operations::

1. Insert

2. Delete

3. Display

4. Exit

Enter Your Choice: 1

Enter the Element: 5

Circular Queue Elements are:3 5

operations::

1. Insert

2. Delete

3. Display

4. Exit

Enter Your Choice: 2

Element 3 is Deleted

Circular Queue Elements are:5

**//Program to implement priority queue**

#include <iostream>

using namespace std;

struct n {

int p;

int info;

struct n \*l;

};

class Priority\_Queue {

private:

n \*f;

public:

Priority\_Queue() {

f = NULL;

}

void insert(int i, int p) {

n \*t, \*q;

t = new n;

t->info = i;

t->p = p;

if (f == NULL || p < f->p) {

t->l= f;

f = t;

} else {

q = f;

while (q->l != NULL && q->l->p <= p)

q = q->l;

t->l = q->l;

q->l = t;

}

}

void del() {

n \*t;

if(f == NULL)

cout<<"Queue Underflow\n";

else {

t = f;

cout<<"Deleted item is: "<<t->info<<endl;

f = f->l;

free(t);

}

}

void show() {

n \*ptr;

ptr = f;

if (f == NULL)

cout<<"Queue is empty\n";

else {

cout<<"Queue is :\n";

cout<<"Priority Item\n";

while(ptr != NULL) {

cout<<ptr->p<<" "<<ptr->info<<endl;

ptr = ptr->l;

}

}

}

};

int main() {

int c, i, p;

Priority\_Queue pq;

do {

cout<<"1.Insert\n";

cout<<"2.Delete\n";

cout<<"3.Display\n";

cout<<"4.Exit\n";

cout<<"Enter your choice : ";

cin>>c;

switch(c) {

case 1:

cout<<"Input the item value to be added in the queue : ";

cin>>i;

cout<<"Enter its priority : ";

cin>>p;

pq.insert(i, p);

break;

case 2:

pq.del();

break;

case 3:

pq.show();

break;

case 4:

break;

default:

cout<<"Wrong choice\n";

}

}

while(c != 4);

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice : 1

Input the item value to be added in the queue : 6

Enter its priority : 2

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice : 1

Input the item value to be added in the queue : 4

Enter its priority : 1

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice : 3

Queue is :

Priority Item

1 4

2 6

**//Program to implement Dequeue**

#include<iostream>

using namespace std;

#define SIZE 10

class dequeue {

int a[20],f,r;

public:

dequeue();

void insert\_at\_beg(int);

void insert\_at\_end(int);

void delete\_fr\_front();

void delete\_fr\_rear();

void show();

};

dequeue::dequeue() {

f=-1;

r=-1;

}

void dequeue::insert\_at\_end(int i) {

if(r>=SIZE-1) {

cout<<"\n insertion is not possible, overflow!!!!";

} else {

if(f==-1) {

f++;

r++;

} else {

r=r+1;

}

a[r]=i;

cout<<"\nInserted item is"<<a[r];

}

}

void dequeue::insert\_at\_beg(int i) {

if(f==-1) {

f=0;

a[++r]=i;

cout<<"\n inserted element is:"<<i;

} else if(f!=0) {

a[--f]=i;

cout<<"\n inserted element is:"<<i;

} else {

cout<<"\n insertion is not possible, overflow!!!";

}

}

void dequeue::delete\_fr\_front() {

if(f==-1) {

cout<<"deletion is not possible::dequeue is empty";

return;

}

else {

cout<<"the deleted element is:"<<a[f];

if(f==r) {

f=r=-1;

return;

} else

f=f+1;

}

}

void dequeue::delete\_fr\_rear() {

if(f==-1) {

cout<<"deletion is not possible::dequeue is empty";

return;

}

else {

cout<<"the deleted element is:"<<a[r];

if(f==r) {

f=r=-1;

} else

r=r-1;

}

}

void dequeue::show() {

if(f==-1) {

cout<<"Dequeue is empty";

} else {

for(int i=f;i<=r;i++) {

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

}

}

}

int main() {

int c,i;

dequeue d;

do {

cout<<"\n 1.insert at beginning";

cout<<"\n 2.insert at end";

cout<<"\n 3.show";

cout<<"\n 4.deletion from front";

cout<<"\n 5.deletion from rear";

cout<<"\n 6.exit";

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

cin>>c;

switch(c) {

case 1:

cout<<"enter the element to be inserted";

cin>>i;

d.insert\_at\_beg(i);

break;

case 2:

cout<<"enter the element to be inserted";

cin>>i;

d.insert\_at\_end(i);

break;

case 3:

d.show();

break;

case 4:

d.delete\_fr\_front();

break;

case 5:

d.delete\_fr\_rear();

break;

case 6:

exit(1);

break;

default:

cout<<"invalid choice";

break;

}

} while(c!=7);

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

1.insert at beginning

2.insert at end

3.show

4.deletion from front

5.deletion from rear

6.exit

enter your choice:1

enter the element to be inserted4

inserted element is:4

1.insert at beginning

2.insert at end

3.show

4.deletion from front

5.deletion from rear

6.exit

enter your choice:1

enter the element to be inserted7

insertion is not possible, overflow!!!

1.insert at beginning

2.insert at end

3.show

4.deletion from front

5.deletion from rear

6.exit

enter your choice:1

enter the element to be inserted7

insertion is not possible, overflow!!!

1.insert at beginning

2.insert at end

3.show

4.deletion from front

5.deletion from rear

6.exit

enter your choice:2

enter the element to be inserted5

Inserted item is 5

**//Program to implement binary tree**

#include <iostream>

using namespace std;

struct node {

node \*l, \*r;

int d;

}\*r = NULL, \*p = NULL, \*np = NULL, \*q;

void create() {

int v,c = 0;

while (c < 6) {

if (r == NULL) {

r = new node;

cout<<"enter value of root node\n";

cin>>r->d;

r->r = NULL;

r->l = NULL;

} else {

p = r;

cout<<"enter value of node\n";

cin>>v;

while(true) {

if (v< p->d) {

if (p->l == NULL) {

p->l = new node;

p = p->l;

p->d = v;

p->l = NULL;

p->r = NULL;

cout<<"value entered in left\n";

break;

} else if (p->l != NULL) {

p = p->l;

}

} else if (v >p->d) {

if (p->r == NULL) {

p->r = new node;

p = p->r;

p->d = v;

p->l = NULL;

p->r = NULL;

cout<<"value entered in right\n";

break;

} else if (p->r != NULL) {

p = p->r;

}

}

}

}

c++;

}

}

**//Program for inorder traversal**

void inorder(node \*p) {

if (p != NULL) {

inorder(p->l);

cout<<p->d<<endl;

inorder(p->r);

}

}

**//Program for preorder traversal**

void preorder(node \*p) {

if (p != NULL) {

cout<<p->d<<endl;

preorder(p->l);

preorder(p->r);

}

}

**//Program for postorder traversal**

void postorder(node \*p) {

if (p != NULL) {

postorder(p->l);

postorder(p->r);

cout<<p->d<<endl;

}

}

int main() {

create();

cout<<" traversal in inorder\n";

inorder(r);

cout<<" traversal in preorder\n";

preorder(r);

cout<<" traversal in postorder\n";

postorder(r);

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tree2.cpp -o tree2 } ; if ($?) { .\tree2 }

enter value of root node

5

enter value of node

3

value entered in left

enter value of node

7

value entered in right

enter value of node

6

value entered in left

enter value of node

8

value entered in right

enter value of node

4

value entered in right

traversal in inorder

3

4

5

6

7

8

traversal in preorder

5

3

4

7

6

8

traversal in postorder

4

3

6

8

7

5

**//Program to insert a node in binary tree**

#include <bits/stdc++.h>

#include<iostream>

using namespace std;

struct Node

{

int data;

Node \*left, \*right;

};

Node\* newNode(int data)

{

Node\* temp = new Node();

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

Node\* Tree(Node\* temp, int data )

{

if (temp == NULL)

return newNode(data);

if (data < temp->data)

temp->left = Tree(temp->left, data);

else

temp->right = Tree(temp->right, data);

return temp;

}

void display(struct Node\* root)

{

if (!root)

return;

display(root->left);

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

display(root->right);

}

void insert(struct Node\* root , int value)

{

queue<struct Node\*> q;

q.push(root);

while (!q.empty()) {

struct Node\* root = q.front();

q.pop();

if (!root->left) {

root->left = newNode(value);

break;

} else

q.push(root->left);

if (!root->right) {

root->right = newNode(value);

break;

} else

q.push(root->right);

}

}

int main()

{

char ch;

int n, arr[20],size;

Node \*root = new Node;

root = NULL;

cout<<"Enter the size : ";

cin>>size;

cout<<"Enter the elements : ";

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

{

cin>>arr[i];

}

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

{

root = Tree(root, arr[i]);

}

cout<<"\nEnter the Element to be insert : ";

cin>>n;

insert(root,n);

cout<<"\nElement Inserted"<<endl;

cout<<"\nAfter Inserting "<<endl;

cout<<"Elements are: ";

display(root);

cout<<endl;

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ treeinsertion.cpp -o treeinsertion } ; if ($?) { .\treeinsertion }

Enter the size : 5

Enter the elements : 3

4

5

3

5

Enter the Element to be insert :

6

Element Inserted

After Inserting

Elements are: 6 3 3 4 5 5

**//Program to delete a node in binary tree**

#include <bits/stdc++.h>

using namespace std;

struct Node {

int key;

struct Node \*left, \*right;

};

struct Node\* newNode(int key)

{

struct Node\* temp = new Node;

temp->key = key;

temp->left = temp->right = NULL;

return temp;

};

void inorder(struct Node\* temp)

{

if (!temp)

return;

inorder(temp->left);

cout << temp->key << " ";

inorder(temp->right);

}

void deletDeepest(struct Node\* root,

struct Node\* d\_node)

{

queue<struct Node\*> q;

q.push(root);

struct Node\* temp;

while (!q.empty()) {

temp = q.front();

q.pop();

if (temp == d\_node) {

temp = NULL;

delete (d\_node);

return;

}

if (temp->right) {

if (temp->right == d\_node) {

temp->right = NULL;

delete (d\_node);

return;

}

else

q.push(temp->right);

}

if (temp->left) {

if (temp->left == d\_node) {

temp->left = NULL;

delete (d\_node);

return;

}

else

q.push(temp->left);

}

}

}

Node\* deletion(struct Node\* root, int key)

{

if (root == NULL)

return NULL;

if (root->left == NULL && root->right == NULL) {

if (root->key == key)

return NULL;

else

return root;

}

queue<struct Node\*> q;

q.push(root);

struct Node\* temp;

struct Node\* key\_node = NULL;

while (!q.empty()) {

temp = q.front();

q.pop();

if (temp->key == key)

key\_node = temp;

if (temp->left)

q.push(temp->left);

if (temp->right)

q.push(temp->right);

}

if (key\_node != NULL) {

int x = temp->key;

deletDeepest(root, temp);

key\_node->key = x;

}

return root;

}

int main()

{

struct Node\* root = newNode(10);

root->left = newNode(11);

root->left->left = newNode(7);

root->left->right = newNode(12);

root->right = newNode(9);

root->right->left = newNode(15);

root->right->right = newNode(8);

cout << "Inorder traversal before deletion : ";

inorder(root);

int key = 11;

root = deletion(root, key);

cout << endl;

cout << "Inorder traversal after deletion : ";

inorder(root);

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ treedel.cpp -o treedel } ; if ($?) { .\treedel }

Inorder traversal before deletion : 7 11 12 10 15 9 8

Inorder traversal after deletion : 7 8 12 10 15 9

**//Program to implement max heap**

#include <iostream>

using namespace std;

void max\_heap(int \*a, int i, int n)

{

int j, temp;

temp = a[i];

j = 2 \* i;

while (j <= n)

{

if (j < n && a[j+1] > a[j])

j = j + 1;

if (temp > a[j])

break;

else if (temp <= a[j])

{

a[j / 2] = a[j];

j = 2 \* j;

}

}

a[j/2] = temp;

return;

}

void build\_maxheap(int \*a,int n)

{

int i;

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

{

max\_heap(a,i,n);

}

}

int main()

{

int n, i, x;

cout<<"enter no of elements \n";

cin>>n;

int a[20];

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

{

cout<<"enter element"<<(i)<<endl;

cin>>a[i];

}

build\_maxheap(a,n);

cout<<"Max Heap\n";

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

{

cout<<a[i]<<endl;

}

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ maxheap.cpp -o maxheap } ; if ($?) { .\maxheap }

enter no of elements

5

enter element1

3

enter element2

5

enter element3

6

enter element4

3

enter element5

5

Max Heap

6

5

3

3

5

**//Program to implement min heap**

#include <iostream>

using namespace std;

void min\_heap(int \*a,int i,int n)

{

int j, temp;

temp = a[i];

j = 2 \* i;

while (j <= n)

{

if (j < n && a[j+1] < a[j])

j = j + 1;

if (temp < a[j])

break;

else if (temp >= a[j])

{

a[j/2] = a[j];

j = 2 \* j;

}

}

a[j/2] = temp;

return;

}

void build\_minheap(int \*a, int n)

{

int i;

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

{

min\_heap(a,i,n);

}

}

int main()

{

int n, i, x;

cout<<"enter no of elements \n";

cin>>n;

int a[20];

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

{

cout<<"enter element"<<(i)<<endl;

cin>>a[i];

}

build\_minheap(a, n);

cout<<"Min Heap\n";

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

{

cout<<a[i]<<endl;

}

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

enter no of elements

4

enter element1

9

enter element2

5

enter element3

7

enter element4

6

Min Heap

5

6

7

9

**//Program to insert a node in max heap**

#include <iostream>

using namespace std;

#define MAX 100

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

{

int root = (i - 1) / 2;

if (arr[root] > 0) {

if (arr[i] > arr[root]) {

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

heapify(arr, n, root);

}

}

}

void insertNode(int arr[], int& n, int Key)

{

n = n + 1;

arr[n - 1] = Key;

heapify(arr, n, n - 1);

}

void printArray(int arr[], int n)

{

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

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

cout << "\n";

}

int main()

{

int arr[MAX] = { 7, 5, 3, 2, 4 };

int n = 5;

int key = 9;

insertNode(arr, n, key);

printArray(arr, n);

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

9 5 7 2 4 3

**//Program to delete a node in max heap**

#include <iostream>

using namespace std;

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

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

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

largest = l;

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

largest = r;

if (largest != i) {

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

heapify(arr, n, largest);

}

}

void deleteRoot(int arr[], int& n)

{

int lastElement = arr[n - 1];

arr[0] = lastElement;

n = n - 1;

heapify(arr, n, 0);

}

void printArray(int arr[], int n)

{

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

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

cout << "\n";

}

int main()

{

int arr[] = { 10, 5, 3, 2, 4 };

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

deleteRoot(arr, n);

printArray(arr, n);

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

5 4 3 2

**//Program to insert a node in min heap**

#include<iostream>

#include<climits>

using namespace std;

void swap(int \*x, int \*y);

class MinHeap

{

int \*harr;

int cap;

int heap\_size;

public:

MinHeap(int cap);

void MinHeapify(int );

int parent(int i)

{ return (i-1)/2; }

int left(int i)

{ return (2\*i + 1); }

int right(int i)

{ return (2\*i + 2); }

int extractMin();

void decreaseKey(int i, int new\_val);

int getMin()

{ return harr[0]; }

void deleteKey(int i);

void insertKey(int k);

};

MinHeap::MinHeap(int cap)

{

heap\_size = 0;

cap = cap;

harr = new int[cap];

}

void MinHeap::insertKey(int k)

{

if (heap\_size == cap)

{

cout << "\n Could not insertKey\n";

return;

}

heap\_size++;

int i = heap\_size - 1;

harr[i] = k;

while (i != 0 && harr[parent(i)] > harr[i])

{

swap(&harr[i], &harr[parent(i)]);

i = parent(i);

}

}

void MinHeap::decreaseKey(int i, int new\_val)

{

harr[i] = new\_val;

while (i != 0 && harr[parent(i)] > harr[i])

{

swap(&harr[i], &harr[parent(i)]);

i = parent(i);

}

}

int MinHeap::extractMin()

{

if (heap\_size <= 0)

return INT\_MAX;

if (heap\_size == 1)

{

heap\_size--;

return harr[0];

}

int root = harr[0];

harr[0] = harr[heap\_size-1];

heap\_size--;

MinHeapify(0);

return root;

}

**//Program to delete a node in max heap**

void MinHeap::deleteKey(int i)

{

decreaseKey(i, INT\_MIN);

extractMin();

}

void MinHeap::MinHeapify(int i)

{

int l = left(i);

int r = right(i);

int smallest = i;

if (l < heap\_size && harr[l] < harr[i])

smallest = l;

if (r < heap\_size && harr[r] < harr[smallest])

smallest = r;

if (smallest != i)

{

swap(&harr[i], &harr[smallest]);

MinHeapify(smallest);

}

}

void swap(int \*x, int \*y)

{

int temp = \*x;

\*x = \*y;

\*y = temp;

}

int main()

{

MinHeap h(11);

h.insertKey(3);

h.insertKey(2);

h.deleteKey(1);

h.insertKey(15);

h.insertKey(5);

h.insertKey(4);

h.insertKey(45);

cout << h.extractMin() << " ";

cout << h.getMin() << " ";

h.decreaseKey(2, 1);

cout << h.getMin();

return 0;

}

**Output:-**

PS D:\cpp programs\cpp> cd "d:\cpp programs\cpp\" ; if ($?) { g++ min2.cpp -o min2 } ; if ($?) { .\min2 }

2 4 1