Vaibhavi Kansal

9923103057

F2 BATCH

WEEK 0(LAB A)..PRACTICE LAB(ARRAY,LIST)..

Answer 1

#include<iostream>

using namespace std;

int main()

{

double arr[]={1,2,3,4,5,6};

double sum=arr[0]+arr[1]+ arr[2]+arr[3]+arr[4]+arr[5];

int n=6;

cout<<"Average of Numbers="<<sum/n<<endl;

return 0;

}

Answer

(a)

#include <iostream>

using namespace std;

class abc

{

int x;

double y;

};

int main()

{

abc \*o1 = new abc, o2;

cout<<"\nSize of o1 :"<<sizeof(o1);

cout<<"\nSize of o2 :" <<sizeof(o2);

cout<<"\nSize of abc is :"<<sizeof(abc);

return 0;

}

(b)

#include <iostream>

using namespace std;

class abc

{

int x;

double y;

int z;

};

int main()

{

abc \*o1 = new abc, o2;

cout<<"\nSize of o1 :"<<sizeof(o1);

cout<<"\nSize of o2 :" <<sizeof(o2);

cout<<"\nSize of abc is :"<<sizeof(abc);

return 0;

}

(c)

#include <iostream>

using namespace std;

class abc

{

int x;

double y;

int z;

int a;

};

int main()

{

abc \*o1 = new abc, o2;

cout<<"\nSize of o1 : "<<sizeof(o1);

cout<<"\nSize of o2 : "<<sizeof(o2);

return 0;

}

(d)

#include <iostream>

using namespace std;

class abc

{

float x;

char y;

int z;

double a;

};

int main()

{

abc \*o1 = new abc, o2;

cout<<"\nSize of o1 : "<<sizeof(o1);

cout<<"\nSize of o2 : "<<sizeof(o2);

return 0;

}

(e)

#include <iostream>

using namespace std;

class abc

{

char x[5];

double y;

};

int main()

{

abc \*o1 = new abc, o2;

cout<<"\nSize of o1 : "<<sizeof(o1);

cout<<"\nSize of o2 : "<<sizeof(o2);

return 0;

}

(f)

#include <iostream>

using namespace std;

class abc

{

char x[5];

float y[3];

};

int main()

{

abc \*o1 = new abc, o2;

cout<<"\nSize of o1 : "<<sizeof(o1);

cout<<"\nSize of o2 : "<<sizeof(o2);

return 0;

}

Answer 8

(a)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

float \*a;

a =(float \*)malloc(sizeof(int));

a[0] = 4.5;

cout<<a[0];

return 0;

}

(b)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

int \*a;

a =(int \*)malloc(sizeof(float));

a[0] = 5;

cout<<a[0];

return 0;

}

(c)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

int \*a, \*b;

a =(int \*)malloc(sizeof(int));

b =(int \*)malloc(5\*sizeof(int));

cout<<sizeof(a)<< sizeof(b);

return 0;

}

(d)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

int \*a;

a[0] = (int \*)malloc(sizeof(int));

a[0] = 5;

cout<<a[0];

return 0;

}

(e)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

int \*a[5];

a[0] = (int \*)malloc(sizeof(int));

a[0][0] = 5;

cout<<a[0][0];

return 0;

}

(f)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

struct node{int a[10];};

struct node \*n;

n =(struct node \*)malloc(sizeof(struct

node)); cout<<sizeof(n);

return 0;

}

(g)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

int \*a[5];

a[0] = (int

\*)malloc(2\*sizeof(int)); a[0][1]

= 5;

cout<<a[0][1];

return 0;

}

(h)

#include <iostream>

#include <malloc.h>

using namespace std;

int main()

{

int \*a = (int \*)malloc(5\*sizeof(int));

a[0] = 1; a[1] = 2; a[2] = 3; a[3] = 4;

a[4] = 5; delete(a);

cout<<a[0]<<a[1]<<a[2]<<[3]<<a[4];

return 0;

}

WEEK 0(LAB B)..PRACTICE OBJECT CLASS

DS LAB

WEEK0LAB-B

Answer 1

#include <iostream>

using namespace std;

class Sample {

public:

void printText1();

void printText2();

void printValue(int value);

};

void Sample::printText1() {

cout <<"IncludeHelp.com\n"; }

void Sample::printText2() {

cout <<"Let's learn together\n"; }

void Sample::printValue(int value) {

cout <<"value is: "<< value<< "\n"; }

int main()

{

Sample obj;

obj.printText1();

obj.printText2();

obj.printValue(101);

return 0;

}

Answer 2

#include <iostream>

#include <vector>

using namespace std;

// Class to handle addition of natural numbers

class NaturalNumberCalculator {

public:

int add(int a, int b) {

return a + b;

}

};

// Class to represent and add complex numbers

class ComplexNumber {

public:

int real;

int imaginary;

ComplexNumber(int r = 0, int i = 0) : real(r), imaginary(i) {}

ComplexNumber add(const ComplexNumber& other) {

return ComplexNumber(real + other.real, imaginary + other.imaginary);

}

void print() {

cout << real << " + " << imaginary << "i" << endl;

}

};

// Class to handle addition of matrices

class Matrix {

public:

vector<vector<int>> mat;

int rows;

int cols;

Matrix(int r, int c) : rows(r), cols(c) {

mat.resize(r, vector<int>(c));

}

void inputMatrix() {

cout << "Enter elements of the matrix (" << rows << "x" << cols << "):\n";

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

for (int j = 0; j < cols; ++j) {

cin >> mat[i][j];

}

}

}

Matrix add(const Matrix& other) {

Matrix result(rows, cols);

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

for (int j = 0; j < cols; ++j) {

result.mat[i][j] = mat[i][j] + other.mat[i][j];

}

}

return result;

}

void printMatrix() {

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

for (int j = 0; j < cols; ++j) {

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

}

cout << endl;

}

}

};

int main() {

// Part (i): Add two natural numbers

NaturalNumberCalculator nnCalculator;

int num1, num2;

cout << "Enter two natural numbers to add: ";

cin >> num1 >> num2;

cout << "Sum of natural numbers: " << nnCalculator.add(num1, num2) << endl;

// Part (ii): Add two complex numbers

ComplexNumber c1, c2;

cout << "Enter real and imaginary parts of first complex number: ";

cin >> c1.real >> c1.imaginary;

cout << "Enter real and imaginary parts of second complex number: ";

cin >> c2.real >> c2.imaginary;

ComplexNumber cSum = c1.add(c2);

cout << "Sum of complex numbers: ";

cSum.print();

// Part (iii): Add two matrices

int rows, cols;

cout << "Enter number of rows and columns of matrices: ";

cin >> rows >> cols;

Matrix m1(rows, cols), m2(rows, cols);

cout << "Enter elements for first matrix:\n";

m1.inputMatrix();

cout << "Enter elements for second matrix:\n";

m2.inputMatrix();

Matrix mSum = m1.add(m2);

cout << "Sum of matrices:\n";

mSum.printMatrix();

return 0;

}

Answer 3

#include <iostream>

#include <vector>

#include <string>

#include <limits>

using namespace std;

// Class to represent a Vendor

class Vendor {

public:

string name;

string accessory;

double price;

int quantity;

void inputDetails() {

cout << "Enter vendor name: ";

cin >> ws; // to clear any leading whitespace

getline(cin, name);

cout << "Enter accessory name: ";

getline(cin, accessory);

cout << "Enter price: ";

cin >> price;

cout << "Enter quantity: ";

cin >> quantity;

}

void printDetails() const {

cout << "Vendor Name: " << name << endl;

cout << "Accessory: " << accessory << endl;

cout << "Price: " << price << endl;

cout << "Quantity: " << quantity << endl;

}

};

// Function to input accessory details for all vendors

void inputVendorDetails(vector<Vendor>& vendors) {

for (auto& vendor : vendors) {

vendor.inputDetails();

}

}

// Function to print details of all vendors

void printVendorDetails(const vector<Vendor>& vendors) {

for (const auto& vendor : vendors) {

vendor.printDetails();

cout << "---------------------------" << endl;

}

}

// Function to compare prices of a specific accessory across all vendors

void comparePrices(const vector<Vendor>& vendors, const string& accessoryName) {

cout << "Comparing prices for: " << accessoryName << endl;

for (const auto& vendor : vendors) {

if (vendor.accessory == accessoryName) {

cout << vendor.name << " offers " << accessoryName << " at price " << vendor.price << endl;

}

}

}

// Function to find the vendor with maximum quantity of "LAN Cable"

void findMaxQuantityLAN(const vector<Vendor>& vendors) {

string maxVendor;

int maxQuantity =-1;

for (const auto& vendor : vendors) {

if (vendor.accessory == "LAN Cable" && vendor.quantity > maxQuantity) {

maxQuantity = vendor.quantity;

maxVendor = vendor.name;

}

}

if (maxQuantity ==-1) {

cout << "No vendor supplies LAN Cable." << endl;

} else {

cout << "Vendor with maximum quantity of LAN Cable: " << maxVendor << " (" <<

maxQuantity << ")" << endl;

}

}

// Function to find the vendor with the lowest price for "Keyboard"

void findLowestPriceKeyboard(const vector<Vendor>& vendors) {

string minVendor;

double minPrice = numeric\_limits<double>::max();

for (const auto& vendor : vendors) {

if (vendor.accessory == "Keyboard" && vendor.price < minPrice) {

minPrice = vendor.price;

minVendor = vendor.name;

}

}

if (minPrice == numeric\_limits<double>::max()) {

cout << "No vendor supplies Keyboard." << endl;

} else {

cout << "Vendor with the lowest price of Keyboard: " << minVendor << " (" << minPrice << ")"

<< endl;

}

}

int main() {

const int NUM\_VENDORS = 10;

vector<Vendor> vendors(NUM\_VENDORS);

// Input details for all vendors

inputVendorDetails(vendors);

// Print details of all vendors

printVendorDetails(vendors);

// Compare prices for a specific accessory (e.g., "Mouse")

string accessoryToCompare;

cout << "Enter accessory to compare prices: ";

cin >> ws; // to clear any leading whitespace

getline(cin, accessoryToCompare);

comparePrices(vendors, accessoryToCompare);

// Find vendor with maximum quantity of "LAN Cable"

findMaxQuantityLAN(vendors);

// Find vendor with the lowest price for "Keyboard"

findLowestPriceKeyboard(vendors);

return 0;

}

Answer 4

(a)

#include<iostream>

using namespace std;

class Test {

int x;

};

int main()

{

Test t;

cout << t.x;

return 0;

}

(b)

#include<iostream>

using namespace std;

class Empty {};

int main()

{

cout << sizeof(Empty);

return 0;

}

(c)

#include<iostream>

using namespace std;

class Test

{

static int x;

int \*ptr;

int y;

};

int main()

{

Test t;

cout << sizeof(t)<< \"\";

cout << sizeof(Test \*);

}

(d)

#include <iostream>

class Test

{

public:

int i;

void get();

};

void Test::get()

{

std::cout << \"Enter the value of i: \";

std::cin >> i;

}

Test t;

int main()

{

Test t; // local object

t.get();

std::cout << \"value of i in local

t:\"<<t.i<<\'\\n\';

::t.get();

std::cout << \"value of i in global t:

\"<<::t.i<<\'\\n\';

return 0;

}

(e)

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

int rollNo;

string stdName;

float perc;

public:

void setValue()

{

rollNo = 0;

stdName = "None";

perc = 0.0f;

}

void printValue()

{

cout << "Student's Roll No.: " << rollNo

<< "\n";

cout << "Student's Name: " << stdName

<< "\n";

cout << "Student's Percentage: " << perc

<< "\n";

}

};

int main()

{

Student std;

std.setValue();

std.printValue();

return 0;

}

(f)

#include <iostream>

using namespace std;

class Person {};

int main() {

Person per;

cout << "size of per: " << sizeof(per) <<

endl;

return 0;

}

WEEK 1(LAB A)

1. You are given an empty singly linked list. Assume that this list can contain whole numbers

only. Write functions to:

a. Insert ‘n’ number of data in the singly linked list. Insert from the head.

void insert\_from\_head(int n,Node\*head)

{

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

{

int val;

cout<<"Enter value "<<i+1<<":\t";

cin>>val;

Node\*n\_node=new Node(val);

n\_node->next=head;

head=n\_node;

}

}

Practice this question using Virtual Lab

(https://ds1-iiith.vlabs.ac.in/exp/linked-list/singly-linked

list/sllexercise.html)

b. Find the total number of nodes in the linked list, and give their average.

int nodeCount(Node\*head)

{

int count=0;

Node\*temp=head

while(temp!=nullptr)

{

count++;

temp=temp->next;

}

return count;

}

int nodeAvg(Node\*head)

{

int sum=0;

Node\*temp=head;

while(temp!=nullptr)

{

sum+=temp->data;

temp=temp->next;

}

return sum/nodeCount();

}

c. Print first ‘m’ data from the linked list. Assume that ‘m’ is less than ‘n’.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

m=4

Output: {1, 3,-9, 45, 2}

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

m=10

Output: Incorrect value of m

void printM\_elements(List L)

{

int pos=0;

int m;

cout<<”Enter the value of m:\t”;

cin>>m;

Node\*temp=L.head;

if(m<nodeCount(head))

{

cout<<endl<<”Incorrect value of m”<<endl;

break;

}

while(pos<m)

{

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

temp=temp->next;

pos++;

}

}

d. Find the middle element of the linked list and check if it’s odd or even. Print an

appropriate output.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

Output: 3 is odd

void checkMiddle(Node\*head)

{

if(head=nullptr){cout<<"The list is empty"<<endl;}

Node\*mid=head;

Node\*two=head;

while(two!=nullptr&&two->next!=nullptr)

{

mid=mid->next;

two=two->next->next;

}

switch(mid->data%2)

{

case 0:

cout<<"The middle element of the list ("<<mid->data<<") is even"<<endl;

break;

case 1:

cout<<"The middle element of the list ("<<mid->data<<") is odd"<<endl;

break;

}

}

e. Find the ‘l’ number from the end of the list.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

l=3

Output: {56, 100,-67}

void showL\_from\_head(int l,Node\*head)

{

Node\*temp=head;

int start=1;

while(start<nodeCount()-l+1)

{

temp=temp->next;

start++;

}

while(temp!=nullptr)

{

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

temp=temp->next;

}

}

f.

Find if a given number exists in the list. If it does, write a function to delete it.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

Number to be found: 45

Output: 45 exists in the original list

Final list: {1, 3,-9, 2, 3, 56, 100,-67}

If the value exists multiple times, delete only the first instance.

void deleteNode(int x,Node\*head)

{

Node\*temp=head;

if(head->data==x)

{

head=head->next;

}

while(temp->next!=nullptr)

{

if(temp->next->data==x)

{

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

break;

}

temp=temp->next;

}

showL\_from\_head(nodeCount(head),head);

}

g. Interchange a pair of values with another given pair in the linked list.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

Pairs to be exchanged: {1,3} with {56,100}

Output: {56, 100,-9, 45, 2, 3, 1, 3,-67}

Hint: first check if the pair exists and then apply an interchange function. If multiple

or duplicate pairs are found, consider the first instance of the pair.

void swapPairs(Node\*head,int\*arr1,int\*arr2)

{

node\*temp1=head;

node\*temp2=head;

while(temp1!=nullptr)

{

if(temp1->data==arr1[0]&&temp1->next->data==arr1[1])

{

break;

temp1=temp1->next;

}else{cout<<"One of the pairs doesn't exist"<<endl;}

}

while(temp2!=nullptr)

{

if(temp2->data==arr2[0]&& temp2->next->data==arr2[1])

{

break;

temp2=temp2->next;

}else{cout<<"One of the pairs doesn't exist"<<endl;}

}

if(temp1!=nullptr&&temp2!=nullptr)

{

swap(temp1->data,temp2->data);

swap(temp1->next->data,temp2->next->data);

}

showL\_from\_head(nodeCount(head),head);

}

h. Check whether a given sub-list exists in the given linked list. If it exists, give its

position (i.e., the starting position of the sub-list in the master linked list).

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

Sub-list to be: {3,-9, 45}

Output: Exists at position 2.

{1, 3,-9, 45, 2, 3, 56, 100,-67}

Assumption: consider only the first occurrence of the sub-list.

void findSubList(Node\*head,Node\*subHead)

{

if(subHead==nullptr){cout<<"The sublist is empty"<<endl;}

int pos=1;

Node\*curr= head;

while(curr!=nullptr)

{

Node\*temp1=current;

Node\*temp2=subHead;

while(temp1!=nullptr&&temp2!=nullptr&&temp1->data==temp2->data)

{

temp1=temp1->next;

temp2=temp2->next;

}

if(temp2==nullptr){cout<<"The sublist exists at position "<<pos<<endl;}

curr=curr->next;

pos++;

}

cout<<"Sublist not found"<<endl

}

i.

Reverse a sub-list in the given linked list.

Example:

Input:

Linked list: {1, 3,-9, 45, 2, 3, 56, 100,-67}

Sub-list to be reversed: {3,-9, 45}

Output: {1, 45,-9, 3, 2, 3, 56, 100,-67}

Assume that the user inputs the sub-list is found in the master linked list.

void reverseSubList(Node\*head,Node\*subHead,Node\*subTail)

{

if(subHead==subTail){cout<<"Only one element found"<<endl;}

Node\*prev=nullptr;

Node\*curr=head;

while(curr!=nullptr&&curr!=subHead)

{

prev=curr;

curr=curr->next;

}

Node\*tailNext=subTail->next;

Node\*prevSub=nullptr;

Node\*currentSub=subHead;

while(currentSub!=tailNext)

{

Node\*next=currentSub->next;

currentSub->next=prevSub;

prevSub=currentSub;

currentSub=next;

}

if(prev!=nullptr){prev->next=prevSub;}

else{head = prevSub;}

subHead->next=tailNext;

}

2. Assume that you have a linked list that can contain strings, i.e., each node can

contain a string. Write a function to:

a. Print all the nodes in the linked list

void printNodes(Node\*head)

{

Node\*temp=head;

while(temp!=nullptr)

{

getline(cin,temp->str);

cout<<endl;

temp=temp->next;

}

}

b. Print all the strings (node values) that start with a particular alphabet.

void printChar(Node\*head,char a)

{

Node\*temp=head;

while(temp!=nullptr)

{

if((temp->str)[0]==a)

{

getline(cin,temp->str);

cout<<endl;

temp=temp->next;

}

}

}

c. Find if a given string exists in the linked list or not. Give appropriate output

message.

void findStr(Node\*head,string s)

{

Node\*temp=head;

while(temp!=nullptr)

{

if(strcmp(temp->str,s)==0){cout<<"The string exists"<<endl;}

}

cout<<"The string doesn't exist"<<endl;

}

d. Find the string with maximum length.

int maxStrLength(Node\*head)

{

int len=head->data.length();

Node\*curr=head->next;

while(temp!=nullptr)

{

if(len<temp->str.length()){len=temp->str.length()}

temp=temp->next;

}

return len;

}

e. Find if a node contains “xyz” as a sub-string or not. Give appropriate output

messages.

void checkSubstr(Node\*head)

{

Node\*curr=head;

string substr="xyz";

while(curr!=nullptr)

{

if(curr->str.find(substr)!=string::npos)

{cout<<"Node with string \""<<curr->str<<"\" contains the substring

\""<<substr<<"\""<<endl;}

curr=curr->next;

}

cout<<"No node contains the substring \""<<substr<<"\""<<endl;

}

f.

Interchange the strings given in the positions p1, p2. These positions are user input.

Check conditions that both p1 and p2 positions exist in the given linked list, eg:

suppose that your linked list consists of 4 strings only, and if the user gives p1=7, p2

= 10, then an error message must be generated.

void swapStrings(Node\*head)

{

int p1,p2;

cout<<"Enter position 1:\t";

cin>>p1;

cout<<"Enter position p2:\t";

cin>>p2;

if(p1==p2){cout<<"Both positions are the same,no need to interchange."<<endl;}

Node\*prev1=nullptr,\*n1=head;

Node\*prev2=nullptr,\*n2=head;

int curr=0;

while(n1!=nullptr&&curr!=p1)

{

prev1=n1;

n1=n1->next;

curr++;

}

curr=0;

while(n2!=nullptr&&curr!=p2)

{

prev2=n2;

n2=n2->next;

curr++;

}

if(n1==nullptr||n2==nullptr){cout<<"One or both positions are invalid."<<endl;}

if(prev1!=nullptr){prev1->next=n2;}

else{head=n2;}

if(prev2!=nullptr){prev2->next=n1;}

else{head=n1;}

Node\*temp=n2->next;

n2->next=n1->next;

n1->next=temp;

cout<<"The strings at positions "<<p1<<" and "<<p2<<" have been

interchanged."<<endl;

}

g. Delete a given node (either by value or by position).

void deleteByVal(Node\*head,const string&s)

{

if(head==nullptr){cout<<"The list is empty."<<endl;}

if(strcmp(head->str,s)==0)

{

Node\*temp=head;

head=head->next;

delete temp;

cout<<"The node with the value \""<<s<<"\" is deleted successfully"<<endl;

}

Node\*curr=head;

while(curr->next!=nullptr&&strcmp(curr->next->str,s)!=0) {curr=curr->next;}

if(curr->next==nullptr){cout<<"Value \""<<s<<"\" not found in the list"<<endl;}

Node\*temp=curr->next;

curr->next=curr->next->next;

delete temp;

cout<<"The node with the value \""<<value<<"\" is deleted successfully"<<endl;

}

void deleteByPos(Node\*head,int p)

{

if(head==nullptr){cout<<"The list is empty"<<endl;}

if(p==0)

{

}

Node\*temp=head;

head=head->next;

delete temp;

cout<<"The node at position "<<p<<" is deleted successfully"<<endl;

Node\*curr=head;

for(int i=0;curr!=nullptr&&i<p-1;++i){curr=curr->next;}

if(curr==nullptr||curr->next==nullptr){cout<<"Position "<<p<<" does not

exist"<<endl;}

Node\*temp=curr->next;

curr->next=curr->next->next;

delete temp;

cout<<"The node at position "<<p<<" is deleted successfully"<<endl;

}

3. Implement a circular linked list that can contain integer elements.

class Node

{

public:

int data;

Node\*next;

Node(){}

Node(int x):data(x),next(nullptr){}

Node(int x,Node\*a):data(x),next(a){}

};

class List

{

Node\*head;

public:

List():head(nullptr){}

~List()

{

}

};

if(head!=nullptr)

{

Node\*curr=head;

Node\*nex;

do

{

nex=curr->next;

delete curr;

curr=nex;

}while(curr!=head);

}

a. Insert elements.

void insertNode(int val)

{

Node\*n\_node=new Node(val);

if(head==nullptr)

{

head=n\_node;

head->next=head;

}

else

{

Node\*temp=head;

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

temp->next=n\_node;

n\_node->next=head;

}

}

Practice this question using Virtual Lab

(https://ds1-iiith.vlabs.ac.in/exp/linked-list/circular-linked

list/cllpractice.html)

b. Print elements.

void printList()

{

Node\*temp=head->next;

cout<<head->data<<",";

while(temp!=head)

{

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

temp=temp->next;

}

}

c. Count the number of elements.

int nodeCount()

{

int count=1;

Node\*temp=head->next;

while(temp!=head)

{

count++;

temp=temp->next;

}

return count;

}

d. Find if any element has a negative value.

void findNeg()

{

if(head!=nullptr){cout<<"The list is empty"<<endl;}

Node\*temp=head;

do

{

if(temp->data<0){cout<<"Negative element found"<<endl;}

temp=temp->next;

}while(temp!=head);

cout<<"No negative element found"<<endl;

}

e. Find the number of nodes having a value greater than 15.

int more\_than15()

{

int count=0;

if(head!=nullptr){return 0;}

Node\*temp=head;

do

{

if(temp->data>15){count++;}

temp=temp->next;

}while(temp!=head);

return count;

}

f.

Delete a particular element from the list.

void deleteElement(int y)

{

Node\*temp=head;

if(head->data==y)

{

Node\*a=head->next;

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

temp->next=a;

cout<<"The element containing the value "<<y<<" is deleted

successfully"<<endl;

}while(temp->next!=head)

{

if(temp->next->data==y)

{

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

cout<<"The element containing the value "<<y<<" is deleted

successfully"<<endl;

}temp=temp->next;

}

}

g. Update the value of a particular element.

void updateNode(Node\*y)

{

Node\*temp=head;

if(y==head)

{

Node\*a=head->next;

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

temp->next=a;

delete head;

cout<<"Node updated successfully"<<endl;

}

while(temp->next!=y){temp=temp->next;}

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

cout<<"Node updated successfully"<<endl;

}

h. Insert a value at a given position.

void insert\_at\_pos(int p,int x)

{

Node\*n\_node=new Node(x);

if(p==1)

{

if(head==nullptr)

{

head=n\_node;

head->next=head;

cout<<"Operation successful"<<endl;

}else

{

}

}else

{

Node\*temp=head;

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

temp->next=n\_node;

n\_node->next=head;

head=n\_node;

cout<<"Operation successful"<<endl;

Node\*temp=head;

for(int i=1;i<p-1;i++)

{

if(temp->next==head){cout<<"ERROR:\tThe position is out of

bounds"<<endl;}

temp=temp->next;

}

n\_node->next=temp->next;

temp->next=n\_node;

cout<<"Operation successful"<<endl;

}

}

i.

j.

Delete all nodes that have a prime number as their value.

bool checkPrime(int a)

{

if(a<=1){return false;}

for(int i=2;i<=a/2;i++)

{if(a%i==0){return false;}}

return true;

}

void deletePrime()

{

Node\*temp=head;

while(temp->next!=head)

{

while(checkPrime(temp->next->data)){temp->next=temp->next->next;}

temp=temp->next;

}if(checkPrime(head->data))

{

Node\*a=head->next;

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

temp->next=a;

cout<<"All prime elements deleted successfully"<<endl;

}

}

Remove all the nodes from the list which contains Fibonacci data values.

bool checkFib(int a)

{

if(a<0){return false;}

int x=0,y=1;

if(a==x||a==y){return true;}

int z=x+y;

while(z<=a)

{

if(z==a){return true;}

x=y;

y=z;

z=x+y;

}return false;

}

void deleteFib()

{

Node\*temp=head;

while(temp->next!=head)

{

while(checkFib(temp->next->data)){temp->next=temp->next->next;}

temp=temp->next;

}if(checkFib(head->data))

{

Node\*a=head->next;

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

temp->next=a;

cout<<"All Fibonacci elements deleted successfully"<<endl;

}

}

4. Create an empty doubly linked list to store integers. Perform the following by

writing appropriate functions to:

a. Insert and print elements of the list.

Practice this question using Virtual Lab

(https://ds1-iiith.vlabs.ac.in/exp/linked-list/doubly-linked

list/dllexercise.html)

b. Traverse all nodes and check if the value is divisible by a number ‘m’.

c. Delete all the nodes from the list that are greater than the given value ‘x’.

d. Find the number of elements between two duplicate values.

Example:

Input:

Doubly Linked list: {1, 3,-9, 45, 2,-56, 3, 56, 100,-67, 3, 3}

Duplicate element: 3

Output: No. of elements between a pair of ‘3’ = 4.

Assumption: You are considering only the first instance of duplicity, i.e.,

between {3,-9, 45, 2,-56, 3} and not for instances like {3, 56, 100,-67, 3} nor

for {3, 3} or any others.

WEEK 1(LAB B)

Question Number1:

#include <iostream>

using namespace std;

// Node structure for the linked list

struct Node {

int data;

Node\* next;

};

// Function to insert an element at the beginning of the linked list

void insertAtBeginning(Node\*& head, int newData) {

Node\* newNode = new Node();

newNode->data = newData;

newNode->next = head;

head = newNode;

}

// Function to insert an element at a specific location in the linked list

void insertAtPosition(Node\*& head, int position, int newData) {

Node\* newNode = new Node();

newNode->data = newData;

if (position == 0) {

newNode->next = head;

head = newNode;

return;

}

Node\* current = head;

for (int i = 0; i < position - 1 && current != nullptr; ++i) {

current = current->next;

}

if (current != nullptr) {

newNode->next = current->next;

current->next = newNode;

} else {

cout << "Position out of range" << endl;

}

}

// Function to take an integer number, split its digits, and store the digits in a

linked list

void splitNumberIntoDigits(Node\*& head, int number) {

if (number == 0) {

insertAtBeginning(head, 0);

return;

}

while (number > 0) {

int digit = number % 10;

insertAtBeginning(head, digit);

number /= 10;

}

}

// Function to print the linked list

void printList(Node\* node) {

while (node != nullptr) {

cout << node->data << " ";

node = node->next;

}

cout << endl;

}

int main() {

Node\* head = nullptr;

// Inserting elements at the beginning

insertAtBeginning(head, 3);

insertAtBeginning(head, 2);

insertAtBeginning(head, 1);

cout << "Linked list after inserting elements at the beginning: ";

printList(head);

// Inserting an element at a specific position

insertAtPosition(head, 2, 4);

cout << "Linked list after inserting 4 at position 2: ";

printList(head);

// Splitting a number into its digits and storing them in a linked list

Node\* digitList = nullptr;

int number = 12345;

splitNumberIntoDigits(digitList, number);

cout << "Linked list after splitting the number " << number << " into digits: ";

printList(digitList);

return 0;

}

Question Number 2:

#include <iostream>

using namespace std;

// Node structure for the linked list

struct Node {

char data;

Node\* next;

};

// Function to insert an element at the end of the linked list

void insertAtEnd(Node\*& head, char newData) {

Node\* newNode = new Node();

newNode->data = newData;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

return;

}

Node\* last = head;

while (last->next != nullptr) {

last = last->next;

}

last->next = newNode;

}

// Function to remove vowels from the linked list

void removeVowels(Node\*& head) {

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr) {

if (current->data == 'a' || current->data == 'e' || current->data == 'i' ||

current->data == 'o' || current->data == 'u' ||

current->data == 'A' || current->data == 'E' || current->data == 'I' ||

current->data == 'O' || current->data == 'U') {

if (prev == nullptr) {

head = current->next;

} else {

prev->next = current->next;

}

Node\* temp = current;

current = current->next;

delete temp;

} else {

prev = current;

current = current->next;

}

}

}

// Function to print the linked list

void printList(Node\* node) {

while (node != nullptr) {

cout << node->data << " ";

node = node->next;

}

cout << endl;

}

int main() {

Node\* head = nullptr;

string name;

cout << "Enter a name: ";

cin >> name;

// Generating linked list from the name

for (char c : name) {

insertAtEnd(head, c);

}

cout << "Linked list before removing vowels: ";

printList(head);

// Removing vowels from the linked list

removeVowels(head);

cout << "Linked list after removing vowels: ";

printList(head);

return 0;

}

Question Number 3:

#include <iostream>

using namespace std;

// Node structure for the linked list

struct Node {

char data;

Node\* next;

};

// Function to insert an element at the end of the linked list

void insertAtEnd(Node\*& head, char newData) {

Node\* newNode = new Node();

newNode->data = newData;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

return;

}

Node\* last = head;

while (last->next != nullptr) {

last = last->next;

}

last->next = newNode;

}

// Function to print the linked list

void printList(Node\* node) {

while (node != nullptr) {

cout << node->data << " ";

node = node->next;

}

cout << endl;

}

// Function to check if the next three nodes match the given three characters

bool matchNextThree(Node\* node, char c1, char c2, char c3) {

return node != nullptr && node->next != nullptr && node->next->next !=

nullptr &&

node->data == c1 && node->next->data == c2 && node->next->next

>data == c3;

}

// Function to remove three consecutive characters from the first linked list if

they match with the second linked list

void removeMatches(Node\*& head1, Node\* head2) {

if (head2 == nullptr || head2->next == nullptr || head2->next->next ==

nullptr) return;

Node\* ptr2 = head2;

while (ptr2->next->next != nullptr) {

char c1 = ptr2->data;

char c2 = ptr2->next->data;

char c3 = ptr2->next->next->data;

Node\* current = head1;

Node\* prev = nullptr;

while (current != nullptr && current->next != nullptr && current->next

>next != nullptr) {

if (matchNextThree(current, c1, c2, c3)) {

Node\* temp1 = current;

Node\* temp2 = current->next;

Node\* temp3 = current->next->next;

if (prev == nullptr) {

head1 = temp3->next;

} else {

prev->next = temp3->next;

}

current = temp3->next;

delete temp1;

delete temp2;

delete temp3;

} else {

prev = current;

current = current->next;

}

}

ptr2 = ptr2->next;

}

}

int main() {

Node\* head1 = nullptr;

Node\* head2 = nullptr;

char input;

cout << "Enter 10 characters to store a name in the first linked list: ";

for (int i = 0; i < 10; ++i) {

cin >> input;

insertAtEnd(head1, input);

}

cout << "Enter 5 characters to store in the second linked list: ";

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

cin >> input;

insertAtEnd(head2, input);

}

cout << "First linked list before removal: ";

printList(head1);

cout << "Second linked list: ";

printList(head2);

// Removing matches

removeMatches(head1, head2);

cout << "First linked list after removal: ";

printList(head1);

return 0;

}

Question Number 4:

#include <iostream>

using namespace std;

// Node structure for the doubly linked list

struct Node {

int data;

Node\* next;

Node\* prev;

};

// Function to insert a new node at the end of the doubly linked list

void insertAtEnd(Node\*& head, int newData) {

Node\* newNode = new Node();

newNode->data = newData;

newNode->next = nullptr;

newNode->prev = nullptr;

if (head == nullptr) {

head = newNode;

return;

}

Node\* last = head;

while (last->next != nullptr) {

last = last->next;

}

last->next = newNode;

newNode->prev = last;

}

// Function to insert an element at a specific location in the doubly linked list

void insertAtPosition(Node\*& head, int position, int newData) {

Node\* newNode = new Node();

newNode->data = newData;

if (position == 0) {

newNode->next = head;

newNode->prev = nullptr;

if (head != nullptr) {

head->prev = newNode;

}

head = newNode;

return;

}

Node\* current = head;

for (int i = 0; i < position - 1 && current != nullptr; ++i) {

current = current->next;

}

if (current == nullptr) {

cout << "Position out of range" << endl;

delete newNode;

return;

}

newNode->next = current->next;

newNode->prev = current;

if (current->next != nullptr) {

current->next->prev = newNode;

}

current->next = newNode;

}

// Function to print the doubly linked list

void printList(Node\* node) {

while (node != nullptr) {

cout << node->data << " ";

node = node->next;

}

cout << endl;

}

int main() {

Node\* head = nullptr;

// Inserting elements at the end

insertAtEnd(head, 1);

insertAtEnd(head, 2);

insertAtEnd(head, 4);

insertAtEnd(head, 5);

cout << "Doubly linked list before insertion: ";

printList(head);

// Inserting an element at a specific position

insertAtPosition(head, 2, 3);

cout << "Doubly linked list after inserting 3 at position 2: ";

printList(head);

return 0;

}

Question Number 5:

#include <iostream>

using namespace std;

// Node structure for the doubly linked list

struct Node {

int data;

Node\* next;

Node\* prev;

};

// Function to insert a new node at the end of the doubly linked list

void insertAtEnd(Node\*& head, int newData) {

Node\* newNode = new Node();

newNode->data = newData;

newNode->next = nullptr;

newNode->prev = nullptr;

if (head == nullptr) {

head = newNode;

return;

}

Node\* last = head;

while (last->next != nullptr) {

last = last->next;

}

last->next = newNode;

newNode->prev = last;

}

// Function to delete the last element from the doubly linked list

void deleteLast(Node\*& head) {

if (head == nullptr) {

cout << "List is empty, nothing to delete" << endl;

return;

}

if (head->next == nullptr) {

delete head;

head = nullptr;

return;

}

Node\* last = head;

while (last->next != nullptr) {

last = last->next;

}

last->prev->next = nullptr;

delete last;

}

// Function to print the doubly linked list

void printList(Node\* node) {

while (node != nullptr) {

cout << node->data << " ";

node = node->next;

}

cout << endl;

}

int main() {

Node\* head = nullptr;

// Inserting elements at the end

insertAtEnd(head, 1);

insertAtEnd(head, 2);

insertAtEnd(head, 3);

insertAtEnd(head, 4);

insertAtEnd(head, 5);

cout << "Doubly linked list before deletion: ";

printList(head);

// Deleting the last element

deleteLast(head);

cout << "Doubly linked list after deleting the last element: ";

printList(head);

return 0;

}

Question Number 6:

#include <iostream>

using namespace std;

// Node structure for the doubly linked list

struct Node {

int data;

Node\* next;

Node\* prev;

};

// Function to insert a new node at the end of the doubly linked list

void insertAtEnd(Node\*& head, int newData) {

Node\* newNode = new Node();

newNode->data = newData;

newNode->next = nullptr;

newNode->prev = nullptr;

if (head == nullptr) {

head = newNode;

return;

}

Node\* last = head;

while (last->next != nullptr) {

last = last->next;

}

last->next = newNode;

newNode->prev = last;

}

// Function to print the doubly linked list

void printList(Node\* node) {

while (node != nullptr) {

cout << node->data << " ";

node = node->next;

}

cout << endl;

}

// Function to swap values of nodes at extreme pairs

void ExtremeSwap(Node\* head) {

if (head == nullptr) return;

Node\* start = head;

Node\* end = head;

// Move 'end' to the last node

while (end->next != nullptr) {

end = end->next;

}

while (start != end && start->prev != end) {

// Swap the values of the nodes

int temp = start->data;

start->data = end->data;

end->data = temp;

// Move 'start' forward and 'end' backward

start = start->next;

end = end->prev;

}

}

int main() {

Node\* head = nullptr;

int values[] = {1, 2, 3, 4, 5, 6, 7, 8};

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

// Inserting elements into the doubly linked list

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

insertAtEnd(head, values[i]);

}

cout << "Original doubly linked list: ";

printList(head);

// Performing extreme swaps

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

ExtremeSwap(head);

cout << "Doubly linked list after swap " << (i + 1) << ": ";

printList(head);

}

return 0;

}

Question Number 7:

#include <iostream>

using namespace std;

// Node structure for the polynomial

struct Node {

int coeff; // Coefficient of the term

int power; // Power of the term

Node\* next; // Pointer to the next node

};

// Function to create a new node

Node\* createNode(int coeff, int power) {

Node\* newNode = new Node();

newNode->coeff = coeff;

newNode->power = power;

newNode->next = nullptr;

return newNode;

}

// Function to add a node to the end of the list

void appendNode(Node\*& head, int coeff, int power) {

if (!head) {

head = createNode(coeff, power);

} else {

Node\* temp = head;

while (temp->next) {

temp = temp->next;

}

temp->next = createNode(coeff, power);

}

}

// Function to add two polynomials and return the result

Node\* addPolynomials(Node\* poly1, Node\* poly2) {

Node\* result = nullptr;

Node\* p1 = poly1;

Node\* p2 = poly2;

while (p1 && p2) {

if (p1->power == p2->power) {

appendNode(result, p1->coeff + p2->coeff, p1->power);

p1 = p1->next;

p2 = p2->next;

} else if (p1->power > p2->power) {

appendNode(result, p1->coeff, p1->power);

p1 = p1->next;

} else {

appendNode(result, p2->coeff, p2->power);

p2 = p2->next;

}

}

// Add remaining nodes of polynomial 1

while (p1) {

appendNode(result, p1->coeff, p1->power);

p1 = p1->next;

}

// Add remaining nodes of polynomial 2

while (p2) {

appendNode(result, p2->coeff, p2->power);

p2 = p2->next;

}

return result;

}

// Function to display a polynomial

void displayPolynomial(Node\* head) {

Node\* temp = head;

while (temp) {

cout << temp->coeff << "x^" << temp->power;

temp = temp->next;

if (temp) {

cout << " + ";

}

}

cout << endl;

}

int main() {

// Polynomial 1: 3x^3 + 5x^2 + 6

Node\* poly1 = nullptr;

appendNode(poly1, 3, 3);

appendNode(poly1, 5, 2);

appendNode(poly1, 6, 0);

// Polynomial 2: 6x^3 + 2x^2 + 4x + 2

Node\* poly2 = nullptr;

appendNode(poly2, 6, 3);

appendNode(poly2, 2, 2);

appendNode(poly2, 4, 1);

appendNode(poly2, 2, 0);

// Add the two polynomials

Node\* result = addPolynomials(poly1, poly2);

// Display the result

cout << "Polynomial 1: ";

displayPolynomial(poly1);

cout << "Polynomial 2: ";

displayPolynomial(poly2);

cout << "Result: ";

displayPolynomial(result);

return 0;

}

WEEK 2(LAB A)…(stack and queue)

1)

Dry run :

the value of x and y are 4 and 0 respectively.

We first push 7 to the stack/

then we push x , that is , 4 to the stack.

Then we push x + 5 , that is , 4 + 5 that is 9 to the stack.

The value of y would be the top most element of the stack , that is , 9.

Then we pop 9 out of the stack.

then we push x + y , that is , 4 + 9 to the stack , that is 13.

then we push y – 2 , that is , 9 – 2 = 7 to the stack.

Then we push 3 to the stack.

The value of x would be 3.

Output :

x = 3

y = 9

7

13

4

7

2)

#include <iostream>

#include <vector>

using namespace std;

class Stack

{

public:

int len;

std::vector <int> data;

Stack()

{

len = 0;

}

void pop()

{

if (len == 0)

{

cout << "Error ! Cannot pop from an empty stack";

return;

}

data.pop\_back();

len--;

}

void push(int payload)

{

data.push\_back(payload);

len++;

}

int top()

{

return data[len-1];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

};

bool isPrime(int number)

{

int f = 0;

for (int i =1 ; i <= number ; i++)

{

if (number % i == 0)

f++;

}

if (f == 2)

return true;

else

return false;

}

int main()

{

Stack s;

int number;

cout << "Enter the number which you want the prime factors of : " << endl;

cin >> number;

for (int i = 1 ; i <= number ; i++)

s.push(i);

int current;

cout << "The prime factors of the number are : " << endl;

while (!s.isEmpty())

{

current = s.top();

if (number % current == 0 && isPrime(current))

cout << current << endl;

s.pop();

}

return 0;

}

3)

#include <iostream>

#include <vector>

using namespace std;

class Stack

{

public:

int len;

std::vector <int> data;

Stack()

{

}

len = 0;

void pop()

{

if (len == 0)

{

cout << "Error ! Cannot pop from an empty stack";

return;

}

data.pop\_back();

len--;

}

void push(int payload)

{

data.push\_back(payload);

len++;

}

int top()

{

return data[len-1];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

int length()

{

return len;

}

void display()

{

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

cout << data[i] << endl;

cout << endl;

}

};

int main()

{

Stack s\_main;

Stack s1;

Stack s2;

Stack s\_combined;

int n = 12;

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

s\_main.push(i);

cout << "this is the content of the first stack : " << endl;

s\_main.display();

int current;

int c = 0;

int l = s\_main.length();

while (!s\_main.isEmpty())

{

current = s\_main.top();

if (c < l / 2)

s1.push(current);

else

s2.push(current);

s\_main.pop();

c++;

}

cout << "this is the content of s1 stack : " << endl;

s1.display();

cout << "this is the content of the s2 stack : " << endl;

s2.display();

while (l--)

{

if (!s1.isEmpty())

{

s\_combined.push(s1.top());

s1.pop();

}

else

{

s\_combined.push(s2.top());

s2.pop();

}

}

cout << "This is the content of the combined stack : " << endl;

s\_combined.display();

return 0;

}

4)

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Stack

{

public:

int len;

std::vector <int> data;

Stack()

{

}

len = 0;

void pop()

{

if (len == 0)

{

cout << "Error ! Cannot pop from an empty stack";

return;

}

data.pop\_back();

len--;

}

void push(int payload)

{

data.push\_back(payload);

}

len++;

int top()

{

return data[len-1];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

int length()

{

return len;

}

void display()

{

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

cout << data[i] << endl;

cout << endl;

}

};

int main()

{

Stack s;

int number;

int base;

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

cin >> number;

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

cin >> base;

while (number > 0)

{

s.push(number % base);

number = number / base;

}

cout << "This is the number : "<< number << " in base : " << base << endl;

while (!s.isEmpty())

{

}

cout << s.top();

s.pop();

return 0;

}

5)

6)

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Stack

{

public:

int len;

std::vector <int> data;

Stack()

{

}

len = 0;

void pop()

{

if (len == 0)

{

cout << "Error ! Cannot pop from an empty stack";

return;

}

data.pop\_back();

}

len--;

void push(int payload)

{

data.push\_back(payload);

len++;

}

int top()

{

return data[len-1];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

int length()

{

return len;

}

void display()

{

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

cout << data[i] << endl;

cout << endl;

}

};

int main()

{

Stack s;

string data;

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

getline (cin , data);

bool balanced = true;

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

{

char current = data[i];

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

s.push(current);

else if (current == '}' || current == ']' || current == ')')

{

if (s.length() == 0)

{

balanced = false;

break;

}

if (current == '}')

{

if (s.top() == '{')

s.pop();

else

{

balanced = false;

break;

}

}

if (current == ']')

{

if (s.top() == '[')

s.pop();

else

{

balanced = false;

break;

}

}

if (current == ')')

{

if (s.top() == '(')

s.pop();

else

{

balanced = false;

break;

}

}

}

}

if (!balanced || s.length() != 0)

cout << "The given string is not balanced ! " << endl;

else

cout << "the given string is balanced !" << endl;

return 0;

}

7)

#include <iostream>

#include <vector>

#include <string>

using namespace std;

template <typename T>

class Queue

{

public:

int len;

std::vector <T> data;

Queue()

{

len = 0;

}

void enque(T payload)

{

data.push\_back(payload);

len++;

}

void deque()

{

if (len == 0)

{

cout << "Cannot deque from an empty queue !";

return;

}

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

data[i] = data[i+1];

data.pop\_back();

len--;

}

void display()

{

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

cout << data[i];

cout << endl;

}

T rear()

{

return data[len-1];

}

T front()

{

return data[0];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

};

int main()

{

Queue<char> q;

string data;

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

getline(cin , data);

int count = 1;

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

{

char current = data[i];

if (current == ' ')

continue;

if (q.isEmpty())

q.enque(current);

else

{

if (current == q.rear())

count++;

else

{

if (count != 1)

{

q.enque((char) (48+count));

count = 1;

}

}

q.enque(current);

}

}

cout << "this is the content of the queue after the operation : " << endl;

q.display();

}

8)

9)

#include <iostream>

#include <vector>

#include <string>

using namespace std;

template <typename T>

class Stack

{

public:

int len;

std::vector <T> data;

Stack()

{

len = 0;

}

void pop()

{

if (len == 0)

{

cout << "Error ! Cannot pop from an empty stack";

return;

}

data.pop\_back();

len--;

}

void push(T payload)

{

data.push\_back(payload);

len++;

}

int top()

{

return data[len-1];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

int length()

{

return len;

}

void display()

{

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

cout << data[i] << endl;

cout << endl;

}

};

template <typename T>

class Queue

{

public:

int len;

std::vector <T> data;

Queue()

{

}

len = 0;

void enque(T payload)

{

data.push\_back(payload);

len++;

}

void deque()

{

if (len == 0)

{

cout << "Cannot deque from an empty queue !";

return;

}

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

data[i] = data[i+1];

data.pop\_back();

len--;

}

void display()

{

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

cout << data[i];

cout << endl;

}

T rear()

{

return data[len-1];

}

T front()

{

return data[0];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

};

int main()

{

Queue<char> q;

Stack <char> s;

string data;

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

getline(cin , data);

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

{

if ((int)data[i] >= 65 && (int)data[i] <= 90)

{

data[i] = (char)((int)data[i] + 32);

}

}

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

{

char current = data[i];

s.push(current);

q.enque(current);

}

bool palindrome = true;

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

{

if (s.top() != q.front())

{

palindrome = false;

break;

}

s.pop();

q.deque();

}

if (palindrome)

cout << "The given data is palindrome in nature ! " << endl;

else

cout << "the given data is not palindrome in nature ! " << endl;

}

10)

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

class Queue

{

public:

int len;

std::vector <T> data;

Queue()

{

}

len = 0;

void enque(T payload)

{

data.push\_back(payload);

len++;

}

void deque()

{

if (len == 0)

{

cout << "Cannot deque from an empty queue !";

return;

}

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

data[i] = data[i+1];

data.pop\_back();

len--;

}

void display()

{

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

cout << data[i];

cout << endl;

}

T rear()

{

return data[len-1];

}

T front()

{

return data[0];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

void moveNthFront(int n)

{

T buffer = data[n];

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

data[i] = data[i-1];

data[0]= buffer;

}

};

int main()

{

Queue<int> q;

int

l = 10;

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

{

q.enque(i + 1);

}

cout << "This is the state of the queue : " << endl;

q.display();

int n ;

cout << "Enter the value of n : " << endl;

cin >> n;

q.moveNthFront(n);

cout << "this is the queue after : " << endl;

q.display();

return 0;

}

8)

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

class Queue

{

public:

int len;

std::vector <T> data;

Queue()

{

}

len = 0;

void enque(T payload)

{

data.push\_back(payload);

len++;

}

void deque()

{

if (len == 0)

{

cout << "Cannot deque from an empty queue !";

return;

}

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

data[i] = data[i+1];

data.pop\_back();

len--;

}

void display()

{

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

cout << data[i] << " ";

cout << endl;

}

T rear()

{

return data[len-1];

}

T front()

{

return data[0];

}

bool isEmpty()

{

if (len == 0)

return true;

else

return false;

}

void moveNthFront(int n)

{

T buffer = data[n-1];

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

data[i] = data[i-1];

data[0]= buffer;

}

};

int main()

{

Queue<int> q;

int

l = 10;

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

{

q.enque(i + 1);

}

cout << "This is the state of the queue : " << endl;

q.display();

int n ;

cout << "Enter the value of n : " << endl;

cin >> n;

q.moveNthFront(n);

cout << "this is the queue after : " << endl;

q.display();

return 0;

}

WEEK 2(LAB B)…STACK AND QUEUES..

Ans 1

#include <iostream>

using namespace std;

struct Node {

char data;

Node\* next;

Node(char d) : data(d), next(nullptr) {}

};

class Stack {

private:

Node\* top;

public:

Stack() : top(nullptr) {}

void push(char data) {

Node\* newNode=newNode(data);

newNode->next = top;

top = newNode;

}

voidpop(){

if(top==nullptr)return;

Node\*temp=top;

top=top->next;

deletetemp;

}

chargetTop(){

if(top==nullptr)return'\0';

returntop->data;

}

boolisEmpty(){

returntop==nullptr;

}

};

boolisValid(strings){

Stackst;

for(charch:s){

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

st.push(ch);

}else{

if(st.isEmpty())returnfalse;

chartop=st.getTop();

st.pop();

if((ch==')'&&top!='(')||

(ch=='}'&&top!='{')||

(ch==']'&&top!='[')){

returnfalse;

}

}

}

returnst.isEmpty();

}

intmain(){

strings="{[()]}";

if(isValid(s)){

cout<<"Thestringisvalid."<<endl;

}else{

cout<<"Thestringisnotvalid."<<endl;

}

return0;

}

Ans2

#include<iostream>

usingnamespacestd;

classstack

{

public:

intarr[10];

inttop=-1;

stack()

{

}

voidpush(intd)

{

top++;

if(top>9)

{

cout<<"stackoverflow:";

return;

}

arr[top]=d;

}

voidpop()

{

if(top<0)

{

cout<<"stackunderflow:";

}

top--;

}

voidfindnextgreater(inta)

{

inti=0,temp=a;

while(temp!=arr[i])

{

i++;

if(i>9)

{

cout<<"elementnotfound:";

return;

}

}

intj=i+1;

if(j>9)

{

cout<<"nonextelement:";

}

while(j<10)

{

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

{

cout<<"elementfound atposition:"<<j-i<<" andtheelementis:

"<<arr[j]<<endl;

return;

}

j++;

}

}

};

intmain()

{

stacks;

inti;

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

{

s.push(i+1);

}

s.findnextgreater(4);

}

Ans3

#include<iostream>

usingnamespacestd;

classstack

{

public:

intarr[10];

inttop=-1;

stack()

{

}

voidpush(intd)

{

top++;

if(top>9)

{

cout<<"stackoverflow:";

return;

}

arr[top]=d;

}

voidpop()

{

if(top<0)

{

cout<<"stackunderflow:";

}

top--;

}

voidfindnextgreater(inta)

{

inti=0,temp=a;

while(temp!=arr[i])

{

i++;

if(i>9)

{

cout<<"elementnotfound:";

return;

}

}

intj=i+1;

while(j!=i)

{

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

{

if(j>i)

{

cout<<"elementfound atposition:"<<j-i<<" andtheelementis:

"<<arr[j]<<endl;

return;

}

else

{

cout<<"elementfound atposition:"<<10-i-j<<" andthe

elementis:"<<arr[j]<<endl;

return;

}

}

j++;

if(j>9)

{

j=0;

}

}

if(i==j)

{

cout<<"nogreaterelementfound:";

}

}

};

intmain()

{

stack s;

int i;

s.push(10);

s.push(6);

s.push(7);

s.push(2);

s.push(5);

s.push(1);

s.push(0);

s.push(4);

s.push(3);

s.push(0);

s.findnextgreater(6);

}

Ans 4

#include <iostream>

#include <unordered\_map>

using namespace std;

struct Node

{

chardata;

intindex;

Node\*next;

Node(chard,inti):data(d),index(i),next(nullptr){}

};

classQueue

{

private:

Node\*front;

Node\*rear;

public:

Queue():front(nullptr),rear(nullptr){}

voidenqueue(chardata,intindex)

{

Node\*newNode=newNode(data,index);

if(rear==nullptr)

{

front=rear=newNode;

return;

}

rear->next=newNode;

rear=newNode;

}

voiddequeue()

{

if(front==nullptr)

return;

Node\*temp=front;

front=front->next;

if(front==nullptr)

rear=nullptr;

deletetemp;

}

Node\*getFront()

{

returnfront;

}

boolisEmpty()

{

returnfront==nullptr;

}

};

intfirstNonRepeatingCharacter(strings)

{

unordered\_map<char,int>charCount;

Queueq;

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

{

charCount[s[i]]++;

q.enqueue(s[i],i);

}

while(!q.isEmpty())

{

Node\*frontNode=q.getFront();

if(charCount[frontNode->data]==1)

{

returnfrontNode->index;

}

q.dequeue();

}

return-1;

}

intmain()

{

strings="geeksforgeeks";

intindex=firstNonRepeatingCharacter(s);

if(index!=-1)

{

cout<<"Thefirstnon-repeatingcharacterisatindex:"<<index<<endl;

}

else

{

cout << "No non-repeating character found." << endl;

}

return 0;

}

WEEK LAB 3(A)…SEARCHING…

1..

Output:

x=3

y=9

7

13

4

7

2)

With STL

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cout<<"Enter the Number: ";

cin>>n;

stack<int> s;

int i = 2;

while (n != 1)

{

if (n % i == 0)

{

s.push(i);

while (n% i == 0)

{

n = n / i;

}

}

i++;

}

cout<<"Prime Factors of given Number are: ";

while (!s.empty())

{

printf("%d ", s.top());

s.pop();

}

return 0;

}

Output:

Without STL

#include <bits/stdc++.h>

using namespace std;

struct Node

{

};

int data;

struct Node\* next;

class Stack

{

private:

Node \*top;

public:

Stack()

{

}

top=NULL;

void push(int data)

{

}

Node\* temp;

temp = new Node();

if (!temp)

{

}

cout << "\error";

return;

temp->data = data;

temp->next = top;

top = temp;

int isEmpty()

{

}

return top == NULL;

int tp()

{

if (!isEmpty())

return top->data;

else

exit(0);

}

void pop()

{

}

Node\* temp;

if (top == NULL)

{

}

cout << "\nStack is empty now" << endl;

return;

else

{

}

temp = top;

top = top->next;

temp->next= NULL;

delete(temp);

void display()

{

Node\* temp;

if (top == NULL)

{

}

cout << "\nStack is empty";

return;

else

{

temp = top;

while (temp != NULL)

{

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

}

temp = temp->next;

}

}

};

int main()

{

Stack s;

cout<<"Enter The Number: ";

int n;

cin>>n;

int i=2;

while (n != 1)

{

}

if (n % i == 0)

{

}

s.push(i);

while (n% i == 0)

{

}

i++;

n = n / i;

cout<<"Prime Factors of given Number are: ";

s.display();

return 0;

}

Output:

3)

#include <bits/stdc++.h>

using namespace std;

int main()

{

cout<<"Enter no of elements: ";

int n;

cin>>n;

stack<int> s,s1,s2,s3;

cout<<"Enter elements for the Stack: ";

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

{

}

int a;

cin>>a;

s.push(a);

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

{

int a=s.top();

s1.push(a);

s.pop();

}

while(!s.empty())

{

}

{

}

int a=s.top();

s2.push(a);

s.pop();

while(!s1.empty())

int a=s1.top();

s3.push(a);

s1.pop();

while(!s2.empty())

{

}

int a=s2.top();

s3.push(a);

s2.pop();

while(!s3.empty())

{

}

cout<<s3.top()<<" ";

s3.pop();

return 0;

}

Output:

4)

#include <bits/stdc++.h>

using namespace std;

int main()

{

cout<<"Enter number: ";

int n;

cin>>n;

int base;

cout<<"enter base: (between 2 to 9): ";

cin>>base;

stack<int> s;

while(n!=0)

{

}

s.push(n%base);

n=n/base;

cout<<"Converted Number: ";

while(!s.empty())

{

}

cout<<s.top();

s.pop();

return 0;

}

Output:

5)

a)

#include <bits/stdc++.h>

using namespace std;

string postfixtoprefix(string s1)

{

stack<string> s;

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

{

char c=s1[i];

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

{

}

string a(1, s1[i]);

s.push(a);

else

{

string op1 = s.top();

s.pop();

string op2 = s.top();

s.pop();

string s2 = s1[i] + op2 + op1;

s.push(s2);

}

}

return s.top();

}

int main()

{

}

cout<<"Enter postfix expression: ";

string s;

cin>>s;

cout<<"Prefix Expression: "<<postfixtoprefix(s);

return 0;

Output:

b)

#include <bits/stdc++.h>

using namespace std;

string prefixtopostfix(string s1)

{

stack<string> s;

for(int i=s1.length()-1; i>=0; i--)

{

char c=s1[i];

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

{

}

string a(1, s1[i]);

s.push(a);

else

{

}

}

string op1 = s.top();

s.pop();

string op2 = s.top();

s.pop();

string s2 = op1+op2 +s1[i];

s.push(s2);

return s.top();

}

int main()

{

cout<<"Enter prefix expression: ";

string s;

cin>>s;

cout<<"Postfix Expression: "<<prefixtopostfix(s);

return 0;

}

Output:

c)

#include <bits/stdc++.h>

using namespace std;

int precision(char c) {

if(c == '^')

return 3;

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

return 2;

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

return 1;

else

return -1;

}

string infixToPostfix(string s)

{

stack<char> st;

string s1;

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

{

char c = s[i];

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

s1 += c;

else if(c == '(')

st.push('(');

else if(c == ')')

{

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

{

}

s1 += st.top();

st.pop();

st.pop();

}

else {

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

s1 += st.top();

st.pop();

}

st.push(c);

}

}

while(!st.empty()) {

s1 += st.top();

st.pop();

}

return s1;

}

int value ( string s)

{

string s1=infixToPostfix(s);

stack<int> st;

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

{

if (s1[i]>='0'&&s1[i]<='9')

{

}

st.push(s1[i]-'0');

else

{

}

}

int val1 = st.top();

st.pop();

int val2 = st.top();

switch (s1[i])

{

case '+': st.push(val2 + val1); break;

case '-': st.push(val2 - val1); break;

case '\*': st.push(val2 \* val1); break;

case '/': st.push(val2/val1); break;

}

return st.top();

}

int main()

{

cout<<"Enter prefix expression: ";

string s;

cin>>s;

cout<<"Postfix Expression: "<<infixToPostfix(s);

string s1="(4+9\*6)-((8-6)/2\*4)\*9/3";

cout<<"\nValue of the expression: (4 + 9 \* 6) - ((8 – 6) / 2 \* 4) \* 9 / 3 : "<<value(s1);

return 0;

}

Output:

6)

#include <bits/stdc++.h>

using namespace std;

bool balance(string s1)

{

stack<char> s;

char x;

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

{

if (s1[i] == '(' || s1[i] == '['

|| s1[i] == '{'|| s1[i]=='/')

{

}

s.push(s1[i]);

if(s1[i]=='/')

i++;

continue;

if (s.empty())

return false;

switch (s1[i])

{

case ')':

x = s.top();

s.pop();

if (x == '{' || x == '['||x=='/')

return false;

break;

case '}':

x = s.top();

s.pop();

if (x == '(' || x == '['||x=='/')

return false;

break;

case ']':

x = s.top();

s.pop();

if (x == '(' || x == '{'||x=='/')

return false;

break;

case '\*':

x = s.top();

i++;

s.pop();

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

return false;

break;

}

}

return (s.empty());

}

int main()

{

}

cout<<"Enter expression: ";

string s;

cin>>s;

if(balance(s))

{

}

cout<<"Yes the expression is balanced";

else

{

}

cout<<"No the expression is not balanced";

return 0;

Output:

7)

#include <bits/stdc++.h>

using namespace std;

void display(queue<char>q)

{

while (!q.empty()) {

cout<< q.front();

q.pop();

}

cout << '\n';

}

queue<char> compress (queue<char>q)

{

queue<char>q1;

while(!q.empty())

{

int count=0;

char c=q.front();

q.pop();

if(c!=' ')

{

q1.push(c);

while(!q.empty())

{

if(q.front()==c)

{

count++;

q.pop();

}

else

{

break;

}

}

if(count>0)

{

q1.push((char)(count+1+48));

}

}

}

return q1;

}

int main()

{

}

queue<char> q;

cout<<"Enter text and end it with '.' :";

while(1)

{

}

char c;

cin>>c;

if(c!='.')

q.push(c);

else

break;

cout << "original text is : ";

display(q);

q=compress(q);

cout<<"compressed text is : ";

display(q);

return 0;

Output:

8)

#include <bits/stdc++.h>

using namespace std;

void display(queue<int>q)

{

}

while (!q.empty()) {

cout<< q.front();

q.pop();

}

cout << '\n';

queue<int> moveNthFront (queue<int>q,int pos)

{

queue<int>q1,q2;

while(!q.empty()&&--pos)

{

}

int c=q.front();

q.pop();

q1.push(c);

if(!q.empty())

{

q2.push(q.front());

q.pop();

while(!q1.empty())

{

q2.push(q1.front());

q1.pop();

}

while(!q.empty())

{

}

q2.push(q.front());

q.pop();

return q2;

}

else

return q1;

}

int main()

{

queue<int> q;

cout<<"Enter queue (press 0 to end)";

while(1)

{

}

int c;

cin>>c;

if(c!=0)

q.push(c);

else

break;

cout << "original queue is : ";

display(q);

cout<<"Enter the position no to move to front : ";

int n;

cin>>n;

q=moveNthFront(q,n);

cout<<"after moving n th element queue is : ";

display(q);

return 0;

}

Output:

9)

#include<bits/stdc++.h>

using namespace std;

bool palindrome (stack<char> s, queue <char>q)

{

while(!s.empty())

{

}

if(s.top()==q.front())

{

}

s.pop();

q.pop();

else

break;

if(s.empty())

{

return true;

}

else

return false;

}

int main()

{

}

cout<<"enter the string:";

string s;

cin>>s;

stack<char> st;

queue<char>q;

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

{

}

st.push(s[i]);

q.push(s[i]);

if(palindrome(st,q))

cout<<"YES it is a palindrome";

else

cout<<"NO it is not a palindrome";

return 0;

Output:..

Week 3 (lab b)…SORTING

ANSWER1

1.

#include <iostream>

#include <vector>

using namespace std;

int findMissingNumber(vector<int>& nums) {

int n = nums.size();

int totalSum = n \* (n + 1) / 2; // Sum of first n natural numbers

int arraySum = 0;

for (int num : nums) {

arraySum += num;

}

return totalSum- arraySum;

}

int main() {

vector<int> nums = {3, 0, 1};

cout << "Missing Number: " << findMissingNumber(nums) << endl;

return 0;

}

2.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

vector<int> twoSum(vector<int>& numbers, int target) {

sort(numbers.begin(), numbers.end());

int left = 0, right = numbers.size()-1;

while (left < right) {

int sum = numbers[left] + numbers[right];

if (sum == target) {

return {left+1, right+1}; // Assuming 1-based indexing

} else if (sum < target) {

left++;

} else {

right--;

}

}

return {}; // Return empty vector if no solution found

}

int main() {

vector<int> numbers = {2, 7, 11, 15};

int target = 9;

vector<int> result = twoSum(numbers, target);

if (!result.empty()) {

cout << "Indices: [" << result[0] << ", " << result[1] << "]" << endl;

} else {

cout << "No pair found that adds up to the target." << endl;

}

return 0;

}

3.

#include <iostream>

#include <vector>

#include <algorithm>

#include <climits>

using namespace std;

vector<pair<int, int>> findMinAbsDifferencePairs(vector<int>& arr) {

sort(arr.begin(), arr.end());

int minDiff = INT\_MAX;

vector<pair<int, int>> result;

for (int i = 1; i < arr.size(); ++i) {

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

if (diff < minDiff) {

minDiff = diff;

result.clear();

result.push\_back({arr[i- 1], arr[i]});

} else if (diff == minDiff) {

result.push\_back({arr[i- 1], arr[i]});

}

}

return result;

}

int main() {

vector<int> arr = {2, 5, 4, 89, 1};

vector<pair<int, int>> pairs = findMinAbsDifferencePairs(arr);

cout << "Pairs with the smallest difference: ";

for (auto& p : pairs) {

cout << "{" << p.first << ", " << p.second << "} ";

}

cout << endl;

return 0;

}

4.

#include <iostream>

using namespace std;

int interpolationSearch(int arr[], int n, int x) {

int lo = 0, hi = n- 1;

while (lo <= hi && x >= arr[lo] && x <= arr[hi]) {

if (lo == hi) {

if (arr[lo] == x) return lo;

return-1;

}

int pos = lo + (((double)(hi- lo) / (arr[hi]- arr[lo])) \* (x- arr[lo]));

if (arr[pos] == x)

return pos;

if (arr[pos] < x)

lo = pos + 1;

else

hi = pos- 1;

}

return-1;

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

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

int x = 4;

int index = interpolationSearch(arr, n, x);

if (index !=-1)

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

else

cout << "Element not found." << endl;

return 0;

}

5.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int minSwaps(vector<int>& nums) {

int n = nums.size();

pair<int, int> arrPos[n];

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

arrPos[i] = {nums[i], i};

}

sort(arrPos, arrPos + n);

vector<bool> visited(n, false);

int ans = 0;

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

if (visited[i] || arrPos[i].second == i)

continue;

int cycleSize = 0;

int j = i;

while (!visited[j]) {

visited[j] = true;

j = arrPos[j].second;

cycleSize++;

}

if (cycleSize > 0) {

ans += (cycleSize- 1);

}

}

return ans;

}

int main() {

vector<int> nums = {4, 3, 2, 1};

cout << "Minimum number of swaps required: " << minSwaps(nums) << endl;

return 0;

}

6.

#include <iostream>

#include <vector>

using namespace std;

int mergeAndCount(vector<int>& arr, int l, int m, int r) {

int leftSize = m- l + 1;

int rightSize = r- m;

vector<int> leftArr(leftSize);

vector<int> rightArr(rightSize);

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

leftArr[i] = arr[l + i];

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

rightArr[i] = arr[m + 1 + i];

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

while (i < leftSize && j < rightSize) {

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

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

} else {

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

swaps += (leftSize- i);

}

}

while (i < leftSize)

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

while (j < rightSize)

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

return swaps;

}

int mergeSortAndCount(vector<int>& arr, int l, int r) {

int count = 0;

if (l < r) {

int m = l + (r- l) / 2;

count += mergeSortAndCount(arr, l, m);

count += mergeSortAndCount(arr, m + 1, r);

count += mergeAndCount(arr, l, m, r);

}

return count;

}

int main() {

vector<int> arr = {1, 20, 6, 4, 5};

int inversionCount = mergeSortAndCount(arr, 0, arr.size()- 1);

cout << "Inversion Count: " << inversionCount << endl;

return 0;

}

WEEK 4(LAB A)…SEARCHING…

Q1

#include <iostream>

void findOccurrences(const int arr[], int size, int key) {

bool found = false;

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

if (arr[i] == key) {

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

found = true;

}

}

if (!found) {

std::cout << "Key " << key << " not found in the array." << std::endl;

}

}

int main() {

int arr[] = {16, 31, 15, 27, 9, 15, 39, 15, 17, 12};

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

int key = 15;

findOccurrences(arr, size, key);

return 0;

}

Q2

#include <iostream>

bool findPairWithProduct(const int arr[], int size, int n) {

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

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

if (arr[i] \* arr[j] == n) {

std::cout << "Pair Found: (" << arr[i] << ", " << arr[j] << ")" << std::endl;

return true;

}

}

}

std::cout << "No pair found with product " << n << std::endl;

return false;

}

int main() {

int arr[] = {5, 20, 3, 2, 50, 80};

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

int n = 150;

findPairWithProduct(arr, size, n);

return 0;

}

Q3

#include <iostream>

#include <algorithm>

void waveSort(int arr[], int size) {

std::sort(arr, arr + size);

for (int i = 0; i < size- 1; i += 2) {

if (i + 1 < size) {

std::swap(arr[i], arr[i + 1]);

}

}

}

int main() {

int arr[] = {10, 90, 49, 2, 1, 5, 23};

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

waveSort(arr, size);

std::cout << "Wave-sorted array: ";

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

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

}

std::cout << std::endl;

return 0;

}

Q4

A)

#include <iostream>

#include <algorithm>

void findOccurrences(int arr[], int size, int key) {

std::sort(arr, arr + size); // Sort the array

int low = 0, high = size- 1, mid;

bool found = false;

while (low <= high) {

mid = low + (high- low) / 2;

if (arr[mid] == key) {

found = true;

break;

}

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

else high = mid- 1;

}

if (!found) {

std::cout << "Key " << key << " not found in the array." << std::endl;

return;

}

int start = mid, end = mid;

while (start >= 0 && arr[start] == key)--start;

while (end < size && arr[end] == key) ++end;

for (int i = start + 1; i < end; ++i) {

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

}

}

int main() {

int arr[] = {5, 20, 3, 2, 50, 80};

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

int key = 15;

findOccurrences(arr, size, key);

return 0;

}

#include <iostream>

bool findPairWithProduct(int arr[], int size, int n) {

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

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

if (arr[i] \* arr[j] == n) {

std::cout << "Pair Found: (" << arr[i] << ", " << arr[j] << ")" << std::endl;

return true;

}

}

}

std::cout << "No pair found with product " << n << std::endl;

return false;

}

int main() {

int arr[] = {5, 20, 3, 2, 50, 80};

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

int n = 150;

findPairWithProduct(arr, size, n);

return 0;

}

#include <iostream>

#include <algorithm>

void waveSort(int arr[], int size) {

std::sort(arr, arr + size);

for (int i = 0; i < size- 1; i += 2) {

if (i + 1 < size) {

std::swap(arr[i], arr[i + 1]);

}

}

}

int main() {

int arr[] = {10, 90, 49, 2, 1, 5, 23};

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

waveSort(arr, size);

std::cout << "Wave-sorted array: ";

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

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

}

std::cout << std::endl;

return 0;

}

WEEK 4(LAB B)….

Q1.]

#include <iostream>

#include <map>

#include <vector>

int main() {

std::vector<int> array = {9, 12, 3, 31, 3, 19, 9, 3};

std::map<int, int> frequency;

// Count the frequency of each element

for (int num : array) {

frequency[num]++;

}

// Print unique elements and their frequencies

std::cout << "Unique: ";

for (const auto& entry : frequency) {

std::cout << entry.first << " ";

}

std::cout << std::endl;

std::cout << "Frequency: ";

for (const auto& entry : frequency) {

std::cout << entry.second << " ";

}

std::cout << std::endl;

return 0;

}

Q2.]

#include <iostream>

#include <cmath> // For sqrt function

int jumpSearch(int arr[], int n, int key) {

// Determine the step size (block size)

int step = std::sqrt(n);

int prev = 0;

// Jump ahead until an element greater than the target is found

while (arr[std::min(step, n)- 1] < key) {

prev = step;

step += std::sqrt(n);

if (prev >= n)

return-1; // Target not found

}

// Perform a linear search within the identified range

while (arr[prev] < key) {

prev++;

if (prev == std::min(step, n))

return-1; // Target not found

}

if (arr[prev] == key)

return prev; // Target found at index prev

return-1; // Target not found

}

int main() {

int array[] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

int size = sizeof(array) / sizeof(array[0]);

int key = 11;

int result = jumpSearch(array, size, key);

if (result !=-1)

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

else

std::cout << "Element not found." << std::endl;

return 0;

}

Q3.]

#include <iostream>

#include <algorithm>

#include <vector>

struct Element {

int count, index, value;

};

bool compareByValue(const Element& a, const Element& b) {

return a.value < b.value;

}

bool compareByFrequency(const Element& a, const Element& b) {

if (a.count != b.count)

return a.count > b.count; // Sort by decreasing frequency

else

return a.index < b.index; // Maintain original order for equal frequencies

}

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

std::vector<Element> elements(n);

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

elements[i].index = i;

elements[i].count = 0;

elements[i].value = arr[i];

}

std::stable\_sort(elements.begin(), elements.end(), compareByValue);

elements[0].count = 1;

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

if (elements[i].value == elements[i- 1].value) {

elements[i].count += elements[i- 1].count + 1;

elements[i- 1].count =-1;

elements[i].index = elements[i- 1].index;

} else {

elements[i].count = 1;

}

}

std::stable\_sort(elements.begin(), elements.end(), compareByFrequency);

for (const auto& e : elements) {

if (e.count !=-1)

std::cout << e.value << " ";

}

}

int main() {

int arr[] = {4, 5, 6, 5, 4, 3};

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

sortByFrequency(arr, n);

return 0;

}

Q4.]

#include <iostream>

#include <vector>

#include <algorithm>

std::vector<int> calculateDifferences(const std::vector<int>& arr) {

std::vector<int> differences;

// Calculate differences between successive elements

for (size\_t i = 1; i < arr.size(); ++i) {

differences.push\_back(std::abs(arr[i]- arr[i- 1]));

}

// Sort the differences in descending order

std::sort(differences.rbegin(), differences.rend());

return differences;

}

int main() {

std::vector<int> arr1 = {4, 1, 3, 5, 4, 3};

std::vector<int> out1 = calculateDifferences(arr1);

std::cout << "Output 1: ";

for (int diff : out1) {

std::cout << diff << " ";

}

std::cout << std::endl;

std::vector<int> arr2 = {3, 1, 4, 5, 4, 3};

std::vector<int> out2 = calculateDifferences(arr2);

std::cout << "Output 2: ";

for (int diff : out2) {

std::cout << diff << " ";

}

std::cout << std::endl;

return 0;

}

WEEK 5(LAB A)….

WEEK5LABA

1)

#include <bits/stdc++.h>

using namespace std;

int search(int arr[], int N, int x)

{

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

if (arr[i] == x)

return i;

return-1;

}

int main(void)

{

int arr[] = { 2, 3, 4, 10, 40 };

int x ;

cin>>x;

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

int result = search(arr, N, x);

(result ==-1)

? cout << "Element is not present in array"

: cout << "Element is present at index " << result;

return 0;

}

2)

#include <bits/stdc++.h>

using namespace std;

int binarySearch(int arr[], int low, int high, int x)

{

while (low <= high) {

int mid = low + (high- low) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] < x)

low = mid + 1;

else

high = mid- 1;

}

return-1;

}

int main(void)

{

int arr[] = { 2, 3, 4, 10, 40 };

int x = 10;

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

int result = binarySearch(arr, 0, n- 1, x);

if(result ==-1) cout << "Element is not present in array";

else cout << "Element is present at index " << result;

return 0;

}

3)

#include <bits/stdc++.h>

using namespace std;

int kthSmallest(int arr[], int N, int K)

{

sort(arr, arr + N);

return arr[K- 1];

}

int main()

{

int arr[] = { 12, 3, 5, 7, 19 };

int K;

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

cout<<"enter the kth smallest element";

cin>>K;

cout << "Kth smallest element is "<< kthSmallest(arr, N, K);

return 0;

}

4)

#include <bits/stdc++.h>

using namespace std;

int interpolationSearch(int arr[], int lo, int hi, int x)

{

int pos;

if (lo <= hi && x >= arr[lo] && x <= arr[hi]) {

pos = lo+ (((double)(hi- lo) / (arr[hi]- arr[lo]))\* (x- arr[lo]));

if (arr[pos] == x)

return pos;

if (arr[pos] < x)

return interpolationSearch(arr, pos + 1, hi, x);

if (arr[pos] > x)

return interpolationSearch(arr, lo, pos- 1, x);

}

return-1;

}

int main()

{

int arr[] = { 10, 12, 13, 16, 18, 19, 20, 21,22, 23, 24, 33, 35, 42, 47 };

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

int x = 18;

int index = interpolationSearch(arr, 0, n- 1, x);

if (index !=-1)

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

else

cout << "Element not found.";

return 0;

}

5)

#include <bits/stdc++.h>

using namespace std;

int binarySearch(string arr[], string x, int n)

{

int l = 0;

int r = n- 1;

while (l <= r) {

int m = l + (r- l) / 2;

int res =-1000;

if (x == (arr[m]))

res = 0;

if (res == 0)

return m;

if (x > (arr[m]))

l = m+1;

Else

r = m-1;

}

return-1;

}

int main()

{

string arr[]= { "contribute", "shrey", "ide", "practice" };

string x;

cin>>x;

int n = 4;

int result = Search(arr, x, n);

if (result ==-1)

cout << ("Element not present");

else

cout << ("Element found at index ") << result;

}

WEEK 5(LAB B)…RECUSRION AND COMPLEXITY

Week 5-LAB B

Q1

#include<iostream>

using namespace std;

bool isPalindrome(int n, int rev = 0) {

if (n == 0) return rev;

rev = rev \* 10 + n % 10;

return isPalindrome(n / 10, rev);

}

int main() {

int n = 121;

if (n == isPalindrome(n)) {

cout << "Palindrome" << endl;

} else {

cout << "Not Palindrome" << endl;

}

return 0;

}

Q2

#include<iostream>

using namespace std;

int sumOfDigits(int n) {

if (n == 0) return 0;

return n % 10 + sumOfDigits(n / 10);

}

int main() {

int n = 1234;

cout << sumOfDigits(n) << endl;

return 0;

}

Q3

#include<iostream>

using namespace std;

int findMax(int arr[], int n) {

if (n == 1) return arr[0];

return max(arr[n- 1], findMax(arr, n- 1));

}

int findMin(int arr[], int n) {

if (n == 1) return arr[0];

return min(arr[n- 1], findMin(arr, n- 1));

}

int main() {

int arr[] = {7, 10, 4, 3, 20, 15};

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

cout << "Maximum: " << findMax(arr, n) << endl;

cout << "Minimum: " << findMin(arr, n) << endl;

return 0;

}

Q4

#include<iostream>

using namespace std;

void reverseString(char str[], int start, int end) {

if (start >= end) return;

char temp = str[start];

str[start] = str[end];

str[end] = temp;

reverseString(str, start + 1, end- 1);

}

int main() {

char str[] = "hello";

int n = sizeof(str) / sizeof(str[0])- 1;

reverseString(str, 0, n- 1);

cout << str << endl;

return 0;

}

Q5

#include<iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* reverseList(Node\* head) {

if (!head || !head->next) return head;

Node\* rest = reverseList(head->next);

head->next->next = head;

head->next = NULL;

return rest;

}

Node\* newNode(int data) {

Node\* node = new Node();

node->data = data;

node->next = NULL;

return node;

}

void printList(Node\* head) {

while (head) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

int main() {

Node\* head = newNode(1);

head->next = newNode(2);

head->next->next = newNode(3);

head->next->next->next = newNode(4);

head = reverseList(head);

printList(head);

return 0;

}

Q6

#include<iostream>

using namespace std;

int gcd(int a, int b) {

if (b == 0) return a;

return gcd(b, a % b);

}

int lcm(int a, int b) {

return (a \* b) / gcd(a, b);

}

int main() {

int a = 12, b = 15;

cout << "GCD: " << gcd(a, b) << endl;

cout << "LCM: " << lcm(a, b) << endl;

return 0;

}

Q7

#include<iostream>

using namespace std;

void permute(int arr[], int l, int r) {

if (l == r) {

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

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

}

cout << endl;

} else {

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

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

permute(arr, l + 1, r);

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

}

}

}

int main() {

int arr[] = {1, 2, 3};

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

permute(arr, 0, n- 1);

return 0;

}

Q8

#include<iostream>

#include<string.h>

using namespace std;

bool isUniqueSubstring(char str[], int start, int end) {

return str[start] == str[end];

}

int countUniqueSubstrings(char str[]) {

int count = 0;

int n = strlen(str);

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

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

if (isUniqueSubstring(str, i, j)) {

count++;

}

}

}

return count;

}

int main() {

char str[] = "andisan";

cout << countUniqueSubstrings(str) << endl;

return 0;

}

WEEK 6 (LAB A)…STL..

Q1)

#include <bits/stdc++.h>

using namespace std;

void insertion\_sort(vector<int> &a)

{

for(int i{};i<a.size();i++)

{

for(int j{1};j<a.size();j++)

{

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

{

swap(a[j],a[j-1]);

}

}

}

}

void disp(vector<int> d)

{

for(int i{};i<d.size();i++)

{

cout<<d[i]<<" ";

}

cout<<endl;

}

int main() {

vector<int> d={3,4,1,6,5,2};

cout<<"Before Sort:"<<endl;

disp(d);

insertion\_sort(d);

cout<<"After Sort:"<<endl;

disp(d);

return 0;

}

Q2)

a)

#include <bits/stdc++.h>

using namespace std;

int freq\_of\_x(vector<int> v,int x)

{

if(find(v.begin(),v.end(),x)==v.end())

{

return 0;

}

unordered\_map<int,int> m;

for(int i{};i<v.size();i++)

{

m[v[i]]++;

}

return m[x];

}

int main() {

vector<int> a={1,2,3,1,2,1,5,6,5,5,1,2,2,1,1,6,7,4,3};

int x;

cout<<"Enter number to get frequency: ";

cin>>x;

cout<<"Frequence of "<<x<<" is: "<<freq\_of\_x(a,x);

return 0;

}

b)

#include <bits/stdc++.h>

using namespace std;

void disp(vector<int> d)

{

for(int i{};i<d.size();i++)

{

cout<<d[i]<<" ";

}

cout<<endl;

}

int main() {

vector<int> a={1,2,3,1,2,1,5,6,5,5,1,2,2,1,1,6,7,4,3};

int x;

cout<<"Enter number to erase: ";

cin>>x;

cout<<"Before:"<<endl;

disp(a);

while(find(a.begin(),a.end(),x)!=a.end())

a.erase(find(a.begin(),a.end(),x));

cout<<"After:"<<endl;

disp(a);

return 0;

}

c)

#include <bits/stdc++.h>

using namespace std;

void erase\_dup(vector<int>& v)

{

map<int,int> m;

for(int i{};i<v.size();i++)

{

m[v[i]]++;

}

v.resize(m.size());

map<int,int>::iterator t=m.begin();

for(int i{};i<v.size();i++,t++)

{

v[i]=t->first;

}

}

void disp(vector<int> d)

{

for(int i{};i<d.size();i++)

{

cout<<d[i]<<" ";

}

cout<<endl;

}

int main() {

vector<int> a={1,2,3,1,2,1,5,6,5,5,1,2,2,1,1,6,7,4,3};

cout<<"Before:"<<endl;

disp(a);

erase\_dup(a);

cout<<"After:"<<endl;

disp(a);

return 0;

}

d)

#include <bits/stdc++.h>

using namespace std;

int dist(vector<int>& v)

{

int maxi=v[0],j=0;

for(int i{};i<v.size();i++)

{

if(v[i]>maxi)

{

maxi=v[i];

j=i;

}

}

return j;

}

int main() {

vector<int> a={30,21,45,31,42,29,67,55};

cout<<"Distance between 1st and max element is: "<<dist(a);

return 0;

}

Q3)

a),b)

#include <bits/stdc++.h>

using namespace std;

int main() {

list<int> l={1,2,3,4,5,6};

cout<<"First element: "<<l.front()<<"\n"<<"Last element: "<<l.back();

return 0;

}

c),d)

#include <bits/stdc++.h>

using namespace std;

void disp(list<int> l)

{

list<int>::iterator it=l.begin();

while(it!=l.end())

{

cout<<\*it++<<" ";

}

cout<<endl;

}

int main() {

list<int> l={1,2,3,4,5};

l.push\_back(6);

l.pop\_front();

disp(l);

return 0;

}

e)

#include <bits/stdc++.h>

using namespace std;

void insert\_at\_pos(list<int>& a,int x,int p)

{

list<int>::iterator it=a.begin();

for(int i{};i<p-1;i++)

{

it++;

}

a.insert(it,x);

}

int main() {

list<int> l={1,2,4,5};

insert\_at\_pos(l,3,3);

list<int>::iterator it=l.begin();

while(it!=l.end())

{

cout<<\*it<<" ";

it++;

}

return 0;

}

f),g),h)

#include <bits/stdc++.h>

using namespace std;

void rem\_ele(list<int>& a,int x)

{

list<int>::iterator it=a.begin();

while(it!=a.end())

{

if(\*it==x)

{

it=a.erase(it);

}

it++;

}

}

int main() {

list<int> l={1,2,1,4,5,1,2};

cout<<"Size of list: "<<l.size()<<endl;

rem\_ele(l,1);

list<int>::iterator it=l.begin();

while(it!=l.end())

{

cout<<\*it<<" ";

it++;

}

cout<<endl;

l.reverse();

list<int>::iterator i=l.begin();

while(i!=l.end())

{

cout<<\*i<<" ";

i++;

}

return 0;

}

i)

#include <bits/stdc++.h>

using namespace std;

void rem\_dup(list<int>& a)

{

list<int>::iterator it=a.begin(),bef=a.begin();

advance(it,1);

while(it!=a.end())

{

if(\*it==\*bef)

{

it=a.erase(it);

}

else

{

bef=it;

it++;

}

}

}

int main() {

list<int> l={1,2,1,1,4,5,1,2,2};

rem\_dup(l);

list<int>::iterator it=l.begin();

while(it!=l.end())

{

cout<<\*it<<" ";

it++;

}

cout<<endl;

return 0;

}

j)

#include <bits/stdc++.h>

using namespace std;

void swap\_lists(list<int>& l1,list<int>& l2)

{

list<int> l3;

l3.assign(l1.begin(),l1.end());

l1.assign(l2.begin(),l2.end());

l2.assign(l3.begin(),l3.end());

}

void disp(list<int> l)

{

list<int>::iterator it=l.begin();

while(it!=l.end())

{

cout<<\*it++<<" ";

}

cout<<endl;

}

int main() {

list<int> l={1,2,3,4,5},l2={6,7,8};

cout<<"Before Swapping:\n";

cout<<"List 1:\n";

disp(l);

cout<<"List 2:\n";

disp(l2);

swap\_lists(l,l2);

cout<<"After Swapping:\n";

cout<<"List 1: "<<endl;

disp(l);

cout<<"List 2: "<<endl;

disp(l2);

return 0;

}

Q4)

#include <bits/stdc++.h>

using namespace std;

void disp(map<int,int> m)

{

map<int,int>::iterator it=m.begin();

while(it!=m.end())

{

cout<<it->first<<" : "<<it->second<<endl;

it++;

}

}

int main() {

map<int,int> m={{1,1},{2,2},{3,3},{4,4},{5,5}};

cout<<"Size of map: "<<m.size()<<endl;

m.insert(pair<int,int>(6,6));

m.erase(3);

disp(m);

return 0;

}