Exams 2

Exercise 1

#include "bits/stdc++.h"

using namespace std;

template <typename T>

int LastOccurrence(vector<T>& data, const T& value){

int n = data.size();

for(int i=0;i<n;i++) if(data[i] == value) return i;

return -1;

}

int main(){

vector<int> v1 = {1,2,3,4,5};

vector<double> v2 = {1.24,43.34,12.2,2.3,1.2};

cout<<LastOccurrence(v1,3)<<endl;

cout<<LastOccurrence(v1,6)<<endl;

cout<<LastOccurrence(v2,2.3)<<endl;

cout<<LastOccurrence(v2,1.4)<<endl;

}

Exercise 2

template<typename T>

void BubbleSort(Node<T> \*root)

{

Node<T> \*ptr;

Node<T> \*end\_ptr = NULL;

if (root == NULL)

return;

do{

ptr = root;

flag= 0;

while (ptr->next != end\_ptr){

if (ptr->data > ptr->next->data)

{

swap(ptr->data, ptr->next->data);

flag = 1;

}

ptr = ptr->next;

}

end\_ptr = ptr;

}

while (flag);

}

Exercise 3

#include <iostream>

using namespace std;

template <typename T>

bool HasCycle(Node<T> \*root)

{

Node<T> \*slow = root;

Node<T> \*fast = root;

while (slow != nullptr && fast != nullptr && (fast->next != nullptr))

{

slow = slow->next;

fast = fast->next->next;

if (slow == fast)

{

return true;

}

}

return false;

}

Exercise 4

#include <iostream>

using namespace std;

string AdjacentDuplicatesRemoval(string str)

{

int flag=0;

while(flag==0)

{

string temp="";

int n = str.size();

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

{

if(((i+1)<n)&&(str[i]!=str[i+1]))

temp=temp+str[i];

else if (i==(n-1))

{

temp=temp+str[i];

}

else

{

char c= str[i];

i++;

while(i<n&&(str[i]==c))

i++;

flag=1;

i--;

}

}

if(flag==1)flag=0;

else if(flag==0)flag=1;

str= temp;

}

return str;

}

int main() {

cout<<AdjacentDuplicatesRemoval("abbaca")<<endl;

cout<<AdjacentDuplicatesRemoval("azxxxzy")<<endl;

cout<<AdjacentDuplicatesRemoval("caaabbbaac")<<endl;

cout<<AdjacentDuplicatesRemoval("acbbcddc")<<endl;

return 0;

}

Exercise 5

#include <iostream>

#include <vector>

using namespace std;

int maxKSum(vector<int>& data, int k) {

if (k < 0 || data.size() < k) {

int sum = 0;

int i = 0;

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

sum += data[i];

return sum;

}

int maxSum = 0, sum = 0;

int i = 0, j = 0;

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

sum += data[i];

maxSum = sum;

for (i = k; i < data.size(); i++) {

sum += data[i] - data[i - k];

if (sum > maxSum)

maxSum = sum;

}

return maxSum;

}

int main()

{

vector<int> emptyData;

cout << "Input: data = [], k = 3" << endl;

cout << "Output:" << maxKSum(emptyData, 3) << endl;

vector<int> data(6);

data[0] = 3;

data[1] = 9;

data[2] = 1;

data[3] = 8;

data[4] = 2;

data[5] = 7;

cout << "Input: data = [3, 9, 1, 8, 2, 7], k = -1" << endl;

cout << "Output:" << maxKSum(data, -1) << endl;

cout << "Input: data = [3, 9, 1, 8, 2, 7], k = 8" << endl;

cout << "Output:" << maxKSum(data, 8) << endl;

cout << "Input: data = [3, 9, 1, 8, 2, 7], k = 3" << endl;

cout << "Output:" << maxKSum(data, 3) << endl;

return 0;

}