**Array Tasks**

**Q1: Write a program to initialize an array of integers and print all the elements.**

#include <iostream>

using namespace std;

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

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

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

}

}

int main(){

int s = 5;

int myArr[] = {23, 7, 43, 32, 75};

printArr(myArr,s);

}

**Q2: Create a program that finds the maximum and minimum values in an array.**

#include <iostream>

using namespace std;

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

int max = INT\_MIN;

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

if(arr[i]>max){

max = arr[i];

}

}

cout<<"Max: "<<max;

}

int main(){

int s = 5;

int myArr[] = {23, 7, 75, 43, 32};

maxEle(myArr,s);

}

**Q3: Write a program that reverses the elements of an array.**

#include <iostream>

using namespace std;

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

int start = 0;

int end = n - 1;

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

}

int main() {

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

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

cout << "Original array: ";

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

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

}

cout << endl;

revArr(arr, n);

cout << "Reversed array: ";

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

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

}

cout << endl;

return 0;

}

**List Tasks**

**Q1: Implement a program to add, remove, and display elements in a linked list.**

#include <iostream>

#include <list>

using namespace std;

int main() {

list<int> myList;

myList.push\_back(10);

myList.push\_back(20);

myList.push\_back(30);

for(int it : myList){

cout<<it<<" ";

}

myList.pop\_back();

cout<<endl<<"After Popping: ";

for(int it : myList){

cout<<it<<" ";

}

}

**Q2: Write a program that searches for an element in a linked list.**

#include <iostream>

#include <list>

using namespace std;

int main() {

list<int> myList;

int searchele;

myList.push\_back(10);

myList.push\_back(20);

myList.push\_back(30);

for(int it : myList){

cout<<it<<" ";

}

cout<<endl<<"Enter Element to search: ";

cin>>searchele;

bool flag = false;

for(int it : myList){

if(searchele==it){

flag = true;

}

}

if(flag==true){

cout<<"Element Found!";

}else{

cout<<"Not Found!";

}

}

**Q3: Create a program to insert an element at the beginning, middle, and end of a linked list.**

#include <iostream>

#include <list>

using namespace std;

int main() {

list<int> myList;

myList.push\_back(10);

myList.push\_back(20);

myList.push\_back(30);

for(int it : myList){

cout<<it<<" ";

}

myList.push\_front(9);

auto it = myList.begin();

advance(it, myList.size() / 2);

myList.insert(it, 11);

cout<<endl<<"After Adding in Front: ";

for(int it : myList){

cout<<it<<" ";

}

}

**Stack Task**

**Q1: Implement a stack using an array and perform push and pop operations.**

#include <iostream>

using namespace std;

class Stack {

private:

int top;

int arr[100];

public:

Stack() {

top = -1;

}

bool isEmpty() {

return (top == -1);

}

bool isFull() {

return (top == 99);

}

void push(int val) {

if (isFull()) {

cout << "Stack Overflow" << endl;

} else {

arr[++top] = val;

}

}

void pop() {

if (isEmpty()) {

cout << "Stack Underflow" << endl;

} else {

cout << "Popped element: " << arr[top--] << endl;

}

}

void display() {

if (isEmpty()) {

cout << "Stack is empty" << endl;

} else {

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

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

}

cout << endl;

}

}

};

int main() {

Stack s;

s.push(10);

s.push(20);

s.push(30);

s.display();

s.pop();

s.display();

return 0;

}

**Q2: Write a program to check if a given string of parentheses is balanced (e.g., “(())” is balanced, but “(()” is not).**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isBalanced(string str) {

stack<char> s;

for (char ch : str) {

if (ch == '(') {

s.push(ch);

} else if (ch == ')') {

if (s.empty()) {

return false;

}

s.pop();

}

}

return s.empty();

}

int main() {

string str = "(())";

if (isBalanced(str)) {

cout << "The string is balanced." << endl;

} else {

cout << "The string is not balanced." << endl;

}

str = "(()";

if (isBalanced(str)) {

cout << "The string is balanced." << endl;

} else {

cout << "The string is not balanced." << endl;

}

return 0;

}

**Q3: Create a stack-based program to reverse a string (push each character and pop to reverse).**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

string revstr(string str) {

stack<char> s;

for (char ch : str) {

s.push(ch);

}

string reversed = "";

while (!s.empty()) {

reversed += s.top();

s.pop();

}

return reversed;

}

int main() {

string str = "Hello, World!";

string reversed = revstr(str);

cout << "Original string: " << str << endl;

cout << "Reversed string: " << reversed << endl;

return 0;

}

**Queues Task**

**Q1: Write a program to check if the elements in a queue form a palindrome. A palindrome reads the same forwards and backwards (e.g., `{1, 2, 3, 2, 1}`).**

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

bool isPalindrome(queue<int> q) {

stack<int> s;

queue<int> originalQueue = q;

while (!q.empty()) {

s.push(q.front());

q.pop();

}

while (!originalQueue.empty()) {

if (originalQueue.front() != s.top()) {

return false;

}

originalQueue.pop();

s.pop();

}

return true;

}

int main() {

queue<int> q;

q.push(1);

q.push(2);

q.push(3);

q.push(2);

q.push(1);

if (isPalindrome(q)) {

cout << "The queue forms a palindrome." << endl;

} else {

cout << "The queue does not form a palindrome." << endl;

}

return 0;

}

**Q2: Implement a simple program that counts the total number of elements in a queue without modifying the queue's order.**

#include <iostream>

#include <queue>

using namespace std;

int countElements(queue<int> q) {

int count = 0;

while (!q.empty()) {

count++;

q.pop();

}

return count;

}

int main() {

queue<int> q;

q.push(1);

q.push(2);

q.push(3);

q.push(4);

cout << "Total number of elements in the queue: " << countElements(q) << endl;

return 0;

}

**Q3: Write a program to simulate a basic ticket queue, where people enter and leave the line in the order they joined.**

#include <iostream>

#include <queue>

#include <string>

using namespace std;

void displayQueue(queue<string> q) {

while (!q.empty()) {

cout << q.front() << " ";

q.pop();

}

cout << endl;

}

int main() {

queue<string> ticketQueue;

ticketQueue.push("Wishaq");

ticketQueue.push("Akbar");

ticketQueue.push("Ali");

cout << "Current queue: ";

displayQueue(ticketQueue);

while (!ticketQueue.empty()) {

cout << ticketQueue.front() << " has been served." << endl;

ticketQueue.pop();

}

cout << "Queue is now empty." << endl;

return 0;

}

**Vectors Task**

**Q1: Write a program to add elements to a vector and display its size and capacity after each insertion.**

#include <iostream>

#include <vector>

using namespace std;

void SizeAndCapacity(vector<int>& vec) {

cout << "Size: " << vec.size() << ", Capacity: " << vec.capacity() << endl;

}

int main() {

vector<int> vec;

vec.push\_back(10);

SizeAndCapacity(vec);

vec.push\_back(20);

SizeAndCapacity(vec);

vec.push\_back(30);

SizeAndCapacity(vec);

vec.push\_back(40);

SizeAndCapacity(vec);

vec.push\_back(50);

SizeAndCapacity(vec);

return 0;

}

**Q2: Implement a program that removes duplicate values from a vector.**

#include <iostream>

#include <vector>

using namespace std;

void remDup(vector<int>& vec) {

vector<int> result;

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

bool isDuplicate = false;

for (int j = 0; j < result.size(); j++) {

if (vec[i] == result[j]) {

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

result.push\_back(vec[i]);

}

}

vec = result;

}

int main() {

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

cout << "Original vector: ";

for (int val : vec) {

cout << val << " ";

}

cout << endl;

remDup(vec);

cout << "Vector after removing duplicates: ";

for (int val : vec) {

cout << val << " ";

}

cout << endl;

return 0;

}

**Q3: Create a program to sort a vector of integers in ascending order**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

vector<int> vec = {5, 2, 9, 1, 5, 6};

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

cout << "Sorted vector: ";

for (int val : vec) {

cout << val << " ";

}

cout << endl;

return 0;

}