WORKSHEET 4



Name: Ananya Amatya

UID: 24000853

Cyber Security

TASK 1

1. STL Container Practice: Write a program using STL containers that: (40 marks)
   1. Uses vector<string> to store names (5 Marks)
   2. Uses map<string, int> to store age against each name (5 Marks)
   3. Implements functions to:
      1. Add new name-age pair (10 marks)
      2. Find all people above certain age (10 marks)
      3. Sort and display names alphabetically (10 marks)

INPUT:

#include <iostream>

#include <vector>

#include <map>

#include <algorithm>

using namespace std;

vector<string> names;

map<string, int> nameAgeMap;

void addPerson(const string& name, int age)

{

names.push\_back(name);

nameAgeMap[name] = age;

}

void displayAboveAge(int ageLimit)

{

cout << "People above age " << ageLimit << endl;

for (const auto& pair : nameAgeMap)

{

if (pair.second > ageLimit)

{

cout << pair.first << " - " << pair.second << " years old"<<endl;

}

}

}

void displaySortedNames()

{

vector<string> sortedNames = names;

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

cout << "Names sorted alphabetically:"<<endl;

for (const string& name : sortedNames)

{

cout << name << endl;

}

}

int main()

{

addPerson("Anu", 20);

addPerson("Kasu", 19);

addPerson("Tina", 44);

addPerson("Rajya", 33);

int ageLimit;

cout << "Enter age to find people above: ";

cin >> ageLimit;

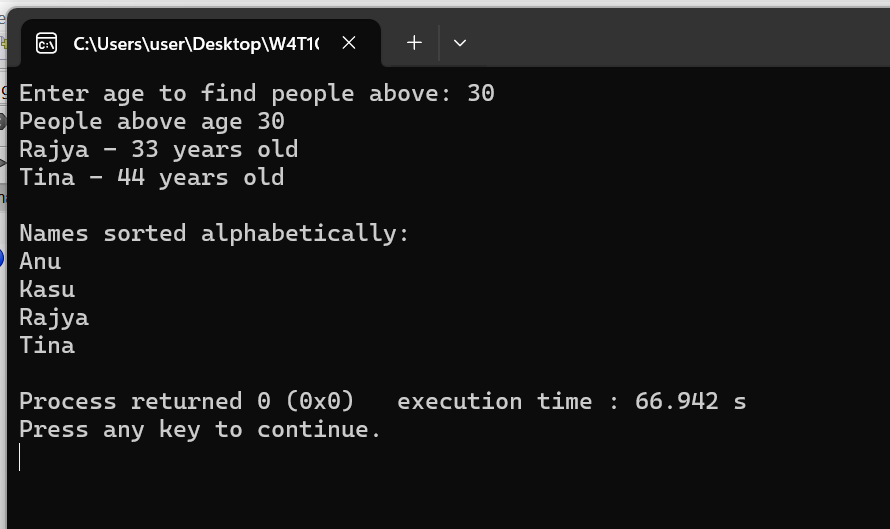
displayAboveAge(ageLimit);

displaySortedNames();

return 0;

}

OUTPUT:



1. Stack Problem: Implement a stack using arrays (not STL) that: (20 marks)
   1. Has basic push and pop operations
   2. Has a function to find middle element
   3. Has a function to reverse only bottom half of stack
   4. Maintain stack size of 10

INPUT:

#include <iostream>

using namespace std;

const int MAX\_SIZE = 10;

class Stack {

int data[MAX\_SIZE];

int top;

public:

Stack() {

top = -1;

}

void push(int value) {

if (top < MAX\_SIZE - 1) {

top++;

data[top] = value;

cout << "Pushed: " << value << endl;

} else {

cout << "Stack is full" << endl;

}

}

void pop() {

if (top >= 0) {

cout << "Popped: " << data[top] << endl;

top--;

} else {

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

}

}

void show() {

cout << "Stack (top to bottom): ";

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

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

}

cout << endl;

}

void findMiddle() {

if (top >= 0) {

int mid = top / 2;

cout << "Middle Element: " << data[mid] << endl;

} else {

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

}

}

void reverseBottomHalf() {

if (top < 1) {

cout << "Not enough elements to reverse bottom half" << endl;

return;

}

int mid = top / 2;

for (int i = 0, j = mid; i < j; i++, j--) {

int temp = data[i];

data[i] = data[j];

data[j] = temp;

}

cout << "Bottom half reversed" << endl;

}

};

int main() {

Stack stack;

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

stack.push(60);

stack.push(70);

stack.push(80);

stack.push(90);

stack.push(100);

stack.show();

stack.findMiddle();

stack.reverseBottomHalf();

stack.show();

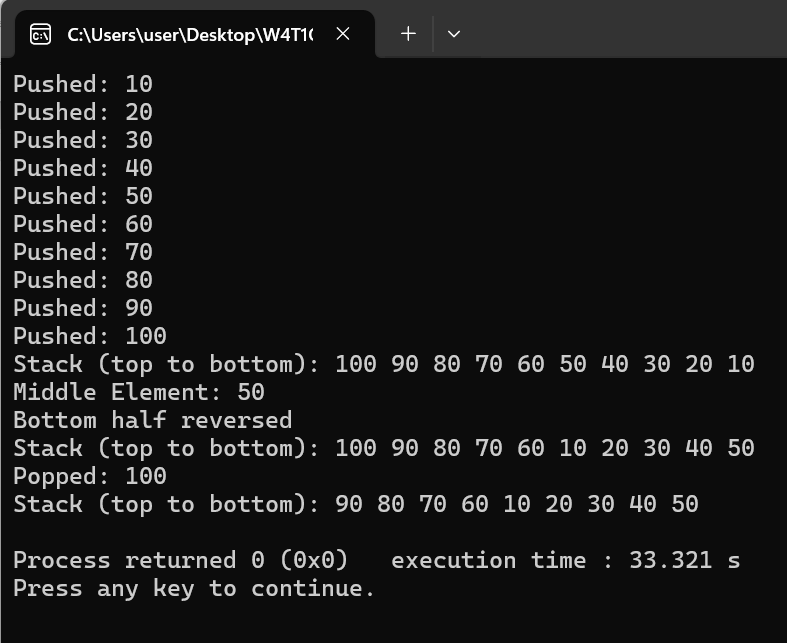
stack.pop();

stack.show();

return 0;

}

OUTPUT:



1. Queue Problem: Implement a queue using arrays (not STL) that: (20 marks)
   1. Has basic enqueue and dequeue operations
   2. Has a function to reverse first K elements
   3. Has a function to interleave first half with second half
   4. Handle queue overflow/underflow

INPUT:

#include <iostream>

using namespace std;

#define MAX\_SIZE 15

class Queue

{

private:

int arr[MAX\_SIZE];

int front, rear, size;

public:

Queue()

{

front = 0;

rear = -1;

size = 0;

}

bool isFull()

{

return size == MAX\_SIZE;

}

bool isEmpty()

{

return size == 0;

}

void enqueue(int val)

{

if (isFull())

{

cout << "Queue overflow! Cannot enqueue " << val << endl;

return;

}

rear = (rear + 1) % MAX\_SIZE;

arr[rear] = val;

size++;

}

int dequeue()

{

if (isEmpty())

{

cout << "Queue underflow! Nothing to dequeue." << endl;

return -1;

}

int val = arr[front];

front = (front + 1) % MAX\_SIZE;

size--;

return val;

}

void display()

{

if (isEmpty())

{

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

return;

}

cout << "Queue: ";

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

{

cout << arr[(front + i) % MAX\_SIZE] << " ";

}

cout << endl;

}

void reverseFirstK(int k)

{

if (k > size || k <= 0)

{

cout << "Invalid value of K."<<endl;

return;

}

int temp[MAX\_SIZE];

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

{

temp[i] = dequeue();

}

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

{

enqueue(temp[i]);

}

int rotate = size - k;

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

{

enqueue(dequeue());

}

cout << "First " << k << " elements reversed."<<endl;

}

void interleaveQueue()

{

if (size % 2 != 0)

{

cout << "Interleave requires even number of elements."<<endl;

return;

}

int half = size / 2;

int temp[MAX\_SIZE];

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

{

temp[i] = dequeue();

}

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

{

enqueue(temp[i]);

enqueue(temp[i + half]);

}

cout << "Queue interleaved."<<endl;

}

};

int main()

{

Queue q;

for (int i = 2; i <= 15; ++i)

q.enqueue(i);

q.display();

q.reverseFirstK(5);

q.display();

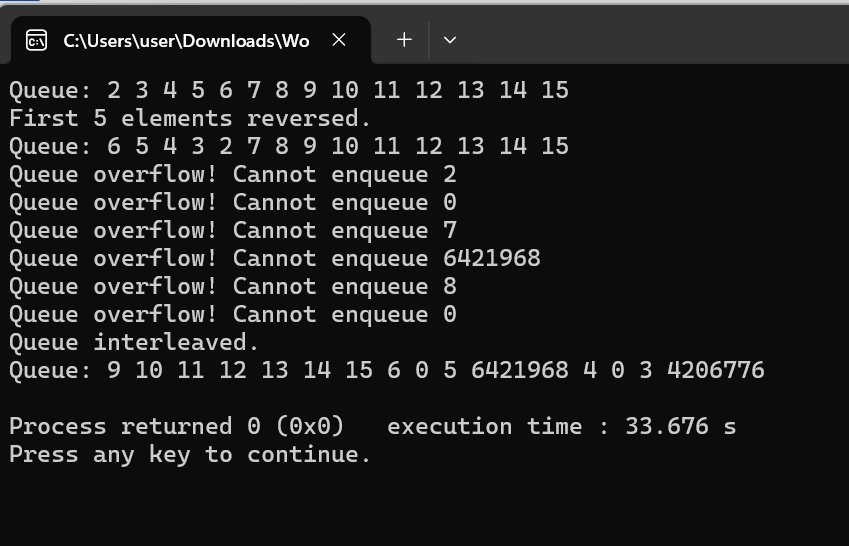
q.interleaveQueue();

q.display();

return 0;

}

OUTPUT:



1. Linked List Problem: Create a singly linked list (not STL) that: (20 marks)
   1. Has functions to insert at start/end/position
   2. Has a function to detect and remove loops
   3. Has a function to find nth node from end
   4. Has a function to reverse list in groups of K nodes

INPUT:

#include <iostream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

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

};

class LinkedList

{

private:

Node\* head;

public:

LinkedList() : head(nullptr) {}

void insertAtStart(int val) {

Node\* newNode = new Node(val);

newNode->next = head;

head = newNode;

}

// 1. Insert at the end

void insertAtEnd(int val) {

Node\* newNode = new Node(val);

if (!head) {

head = newNode;

return;

}

Node\* temp = head;

while (temp->next)

temp = temp->next;

temp->next = newNode;

}

void insertAtPosition(int pos, int val)

{

if (pos == 0)

{

insertAtStart(val);

return;

}

Node\* newNode = new Node(val);

Node\* temp = head;

for (int i = 0; temp && i < pos - 1; ++i)

temp = temp->next;

if (!temp)

{

cout << "Position out of bounds."<<endl;

return;

}

newNode->next = temp->next;

temp->next = newNode;

}

void display()

{

Node\* temp = head;

cout << "Linked List: ";

while (temp)

{

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

temp = temp->next;

}

cout << endl;

}

void detectAndRemoveLoop()

{

Node \*slow = head, \*fast = head;

while (fast && fast->next)

{

slow = slow->next;

fast = fast->next->next;

if (slow == fast)

{

cout << "Removing Loop detected."<<endl;

removeLoop(slow);

return;

}

}

cout << "No loop detected."<<endl;

}

void removeLoop(Node\* loopNode)

{

Node\* ptr1 = head;

Node\* ptr2;

while (true)

{

ptr2 = loopNode;

while (ptr2->next != loopNode && ptr2->next != ptr1)

ptr2 = ptr2->next;

if (ptr2->next == ptr1)

break;

ptr1 = ptr1->next;

}

ptr2->next = nullptr;

}

void findNthFromEnd(int n)

{

Node \*mainPtr = head, \*refPtr = head;

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

{

if (!refPtr)

{

cout << "N is greater than list length."<<endl;

return;

}

refPtr = refPtr->next;

}

while (refPtr)

{

mainPtr = mainPtr->next;

refPtr = refPtr->next;

}

cout << "the node from end is: " << mainPtr->data << endl;

}

Node\* reverseInGroups(Node\* node, int k)

{

Node\* prev = nullptr;

Node\* curr = node;

Node\* next = nullptr;

int count = 0;

while (curr && count < k)

{

next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

count++;

}

if (next)

node->next = reverseInGroups(next, k);

return prev;

}

void reverseGroups(int k)

{

head = reverseInGroups(head, k);

cout << "Reserved list in groups of " << k << endl;

}

void createLoop()

{

if (!head) return;

Node\* temp = head;

while (temp->next)

temp = temp->next;

temp->next = head->next;

}

};

int main() {

LinkedList list;

list.insertAtEnd(10);

list.insertAtEnd(20);

list.insertAtEnd(30);

list.insertAtEnd(40);

list.insertAtEnd(50);

list.insertAtPosition(5, 25);

list.insertAtStart(5);

list.display();

list.findNthFromEnd(3);

list.reverseGroups(3);

list.display();

list.createLoop();

list.detectAndRemoveLoop();

list.display();

return 0;

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

GitHub link: