

**DSA (Data Structure and Algorithms) Lab**

**LAB REPORT # 6**

**Semester**: 3rdSemester

**Section**: C

**Submitted To:**

**Abdullah Shahrose**

**Submitted By:**

**Name**: Faisal Khan

**Roll No**: 22-CS-039

**Task 1:**

#include <iostream>

using namespace std;

struct Node

{

    int data;

    Node \*next;

};

class Stack

{

public:

    Stack() { top = NULL; }

    void push(int data)

    {

        Node \*newNode = new Node();

        newNode->data = data;

        newNode->next = top;

        top = newNode;

    }

    int pop()

    {

        if (top == NULL)

        {

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

            exit(1);

        }

        Node \*temp = top;

        int data = top->data;

        top = top->next;

        delete temp;

        return data;

    }

    bool isEmpty() { return top == NULL; }

    int getMax()

    {

        int max = INT\_MIN;

        Node \*curr = top;

        while (curr != NULL)

        {

            if (curr->data > max)

            {

                max = curr->data;

            }

            curr = curr->next;

        }

        return max;

    }

    int getMin()

    {

        int min = INT\_MAX;

        Node \*curr = top;

        while (curr != NULL)

        {

            if (curr->data < min)

            {

                min = curr->data;

            }

            curr = curr->next;

        }

        return min;

    }

private:

    Node \*top;

};

int main()

{

    Stack stack;

    // Push some elements into the stack

    stack.push(10);

    stack.push(20);

    stack.push(30);

    stack.push(40);

    stack.push(50);

    // Print the maximum and minimum values in the stack

    cout << "Maximum value in the stack: " << stack.getMax() << endl;

    cout << "Minimum value in the stack: " << stack.getMin() << endl;

    return 0;

}

**Output:**



**Task 2:**

#include <iostream>

using namespace std;

struct Node

{

    int data;

    Node \*next;

};

class Stack

{

public:

    Stack() { top = NULL; }

    void push(int data)

    {

        Node \*newNode = new Node();

        newNode->data = data;

        newNode->next = top;

        top = newNode;

    }

    int pop()

    {

        if (top == NULL)

        {

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

            exit(1);

        }

        Node \*temp = top;

        int data = top->data;

        top = top->next;

        delete temp;

        return data;

    }

    bool isEmpty() { return top == NULL; }

    int bestScore()

    {

        int bestScore = INT\_MIN;

        Node \*curr = top;

        while (curr != NULL)

        {

            if (curr->data > bestScore)

            {

                bestScore = curr->data;

            }

            curr = curr->next;

        }

        return bestScore;

    }

    float averageScore()

    {

        float sum = 0;

        int count = 0;

        Node \*curr = top;

        while (curr != NULL)

        {

            sum += curr->data;

            count++;

            curr = curr->next;

        }

        return sum / count;

    }

    int worstScore()

    {

        int worstScore = INT\_MAX;

        Node \*curr = top;

        while (curr != NULL)

        {

            if (curr->data < worstScore)

            {

                worstScore = curr->data;

            }

            curr = curr->next;

        }

        return worstScore;

    }

private:

    Node \*top;

};

int main()

{

    Stack stack;

    // Push the test scores into the stack

    cout << "Enter the test scores: ";

    int score;

    while (cin >> score)

    {

        stack.push(score);

    }

    // Find the best, average, and worst scores

    int bestScore = stack.bestScore();

    float averageScore = stack.averageScore();

    int worstScore = stack.worstScore();

    // Print the results

    cout << "Best score: " << bestScore << endl;

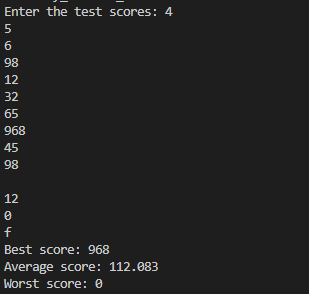
    cout << "Average score: " << averageScore << endl;

    cout << "Worst score: " << worstScore << endl;

    return 0;

}

**Output:**



**Task 3:**

#include <iostream>

using namespace std;

struct Employee

{

    int id;

    string name;

    string department;

    float salary;

};

struct Node

{

    Employee employee;

    Node \*next;

};

class Stack

{

public:

    Stack() { top = NULL; }

    void push(Employee employee)

    {

        Node \*newNode = new Node();

        newNode->employee = employee;

        newNode->next = top;

        top = newNode;

    }

    Employee pop()

    {

        if (top == NULL)

        {

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

            exit(1);

        }

        Node \*temp = top;

        Employee employee = top->employee;

        top = top->next;

        delete temp;

        return employee;

    }

    bool isEmpty() { return top == NULL; }

    void display()

    {

        Node \*curr = top;

        while (curr != NULL)

        {

            cout << curr->employee.id << " " << curr->employee.name << " " << curr->employee.department << " " << curr->employee.salary << endl;

            curr = curr->next;

        }

    }

private:

    Node \*top;

};

int main()

{

    Stack stack;

    // Push some employee records into the stack

    Employee employee1 = {1, "Alice", "Engineering", 100000};

    Employee employee2 = {2, "Bob", "Sales", 90000};

    Employee employee3 = {3, "Carol", "Marketing", 80000};

    stack.push(employee1);

    stack.push(employee2);

    stack.push(employee3);

    // Display the employee records in the stack

    stack.display();

    // Pop an employee record from the stack

    Employee employee = stack.pop();

    // Display the popped employee record

    cout << "Popped employee record: " << endl;

    cout << employee.id << " " << employee.name << " " << employee.department << " " << employee.salary << endl;

    return 0;

}

**Output:**

