NAME : Sinwan Haider

Sapp : 56275

Section :CS3-1

Subject : Data Structure

Lab task 03

QUESTION # 01 :

```
#include <iostream>
#include <conio.h>
using namespace std;
#define MAX_SIZE 100

class Stack {
private:
   int data[MAX_SIZE];
   int top;
public:
   // 1. Constructor: Creates an empty stack
   Stack(int ignored = 0) {
      top = -1;
   }

   // 2. Destructor: Deallocates the memory used to store a stack
   ~Stack() {
      // No need to do anything in C++, as the memory is automatically deallocated
   }

   // 3. Push: Pushes an element to the top of the stack
   void push(const int element) {
      if (top == MAX_SIZE - 1) {
         throw runtime_error("Error: Stack overflow!");
      }
      data[++top] = element;
   }

   // 4. Pop: Returns the element from the top of the stack
   int pop() {
      if (top == -1) {
         throw runtime_error("Error: Stack underflow!");
```

```
        }
        return data[top--];
    }

    // 5. Peek: Returns the element at the top of the stack
    int peek() {
        if (top == -1) {
            throw runtime_error("Error: Stack is empty!");
        }
        return data[top];
    }

    // 6. Clear: Removes all the elements from the stack
    void clear() {
        top = -1;
    }

    // 7. isEmpty: Returns true if the stack is empty, otherwise returns false
    bool isEmpty() {
        return top == -1;
    }
};

int main() {
    Stack stack;
    try {
        stack.push(10);
        stack.push(20);
        cout << "Peek: " << stack.peek() << endl;
        cout << "Pop: " << stack.pop() << endl;
        cout << "IsEmpty: " << (stack.isEmpty() ? "True" : "False") << endl;
        stack.clear();
        cout << "IsEmpty: " << (stack.isEmpty() ? "True" : "False") << endl;
    } catch (const runtime_error& e) {
        cerr << "Error: " << e.what() << endl;
    }
    return 0;
}
```

OUTPUT

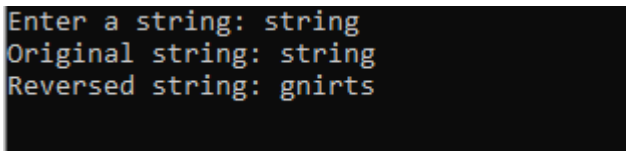
```
Peek: 20
Pop: 20
IsEmpty: False
IsEmpty: True
```

Question no 2

```
#include <iostream>

using namespace std;

#define MAX_SIZE 100

class Stack {
private:
   char data[MAX_SIZE];
   int top;
public:
   Stack() {
      top = -1;
   }

   void push(char element) {
      if (top == MAX_SIZE - 1) {
         cout << "Error: Stack overflow!" << endl;
         exit(1);
      }
      data[++top] = element;
   }

   char pop() {
      if (top == -1) {
         cout << "Error: Stack underflow!" << endl;
         exit(1);
      }
```

```
        return data[top--];
    }
};

void reverse_string(char* str) {
    Stack stack;

    int len = strlen(str);
    for (int i = 0; i < len; i++) {
        stack.push(str[i]);
    }

    for (int i = 0; i < len; i++) {
        str[i] = stack.pop();
    }
}

int main() {
    char str[MAX_SIZE];
    cout << "Enter a string: ";
    cin.getline(str, MAX_SIZE);

    cout << "Original string: " << str << endl;
    reverse_string(str);
    cout << "Reversed string: " << str << endl;

    return 0;
}
```

Output:


Enter a string: string
Original string: string
Reversed string: gnirts