DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

PRACTICAL FILE

Student Name	Onkar Nath
UID	21BCS4159
Section & Group	21BCS_IOT_644_B
Department	Computer Science & Engineering
Session	July-Dec 2023
Course Name	Design and Analysis of Algorithms with Lab
Course Code	21ITH-311/21CSH-311
Semester	5TH

Department of Computer Science & Engineering
Chandigarh University, Mohali

Course Name: DAA Lab **Course Code: 21ITH-311/21CSH-311**

INDEX

S. No.	Experiment	Date	Conduct (12)	Viva (10)	Worksheet (8)	Total (30)	Remarks
1	**Experiment 1.1:** Analyze if stack Isempty, Isfull and if elements are present then return top element in stacks using templates and also perform push and pop operation in stack.						
2	**Experiment 1.2:** Develop a program for implementation of power function and determine that complexity should be O(log n) .						
3	**Experiment 1.3:** Evaluate the complexity of the developed program to find frequency of elements in a given array.						
4	**Experiment 1.4:** i. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and end of Singly Linked List. ii. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.						
5	**Experiment 2.1:** Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random						

	number generator.						
6	**Experiment 2.2** Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.						
7	**Experiment 2.3:** Develop a program and analyze complexity to implement 0-1 Knapsack using Dynamic Programming.						
8	**Experiment 3.1:** Develop a program and analyze complexity to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, OR (ii) to find a path from source to goal in a maze						
9	**Experiment 3.2:** Develop a program and analyze complexity to find shortest paths in a graph with positive edge weights using Dijkstra's algorithm.						
10	**Experiment 3.3:** Develop a program and analyze complexity to find all occurrences of a pattern P in a given string S.						

Student Name: Onkar Nath **UID:** 21BCS4159
Branch:BE-CSE **Section/Group:** 644 B
Semester:5 **Date of Performance:**08-08-2023
Subject Name: DAA **Subject Code:** 21CSH-311

Experiment 1.1

Aim: Analyze if stack Isempty, Isfull and if elements are present then return top element in stacksusing templates and also perform push and pop operation in stack.

Procedure/Algorithm:

```
Step 1: Create Stack: Set up a container to store elements.
Step 2: Check if Stack is Empty:
        If there are no elements in the container, the stack is empty.
Step 3: Check if Stack is Full:
        If the number of elements reaches a limit, the stack is full.
Step 4: Push Element onto Stack:
         If the stack is not full:
        Add an element to the container.
        This becomes the new top element.
Step 5: Pop Element from Stack:
         If the stack is not empty:
         Remove the top element from the container.
        The element below it becomes the new top.
Step 6: Get Top Element:
        If the stack is not empty:
        Return the top element without removing it.
```

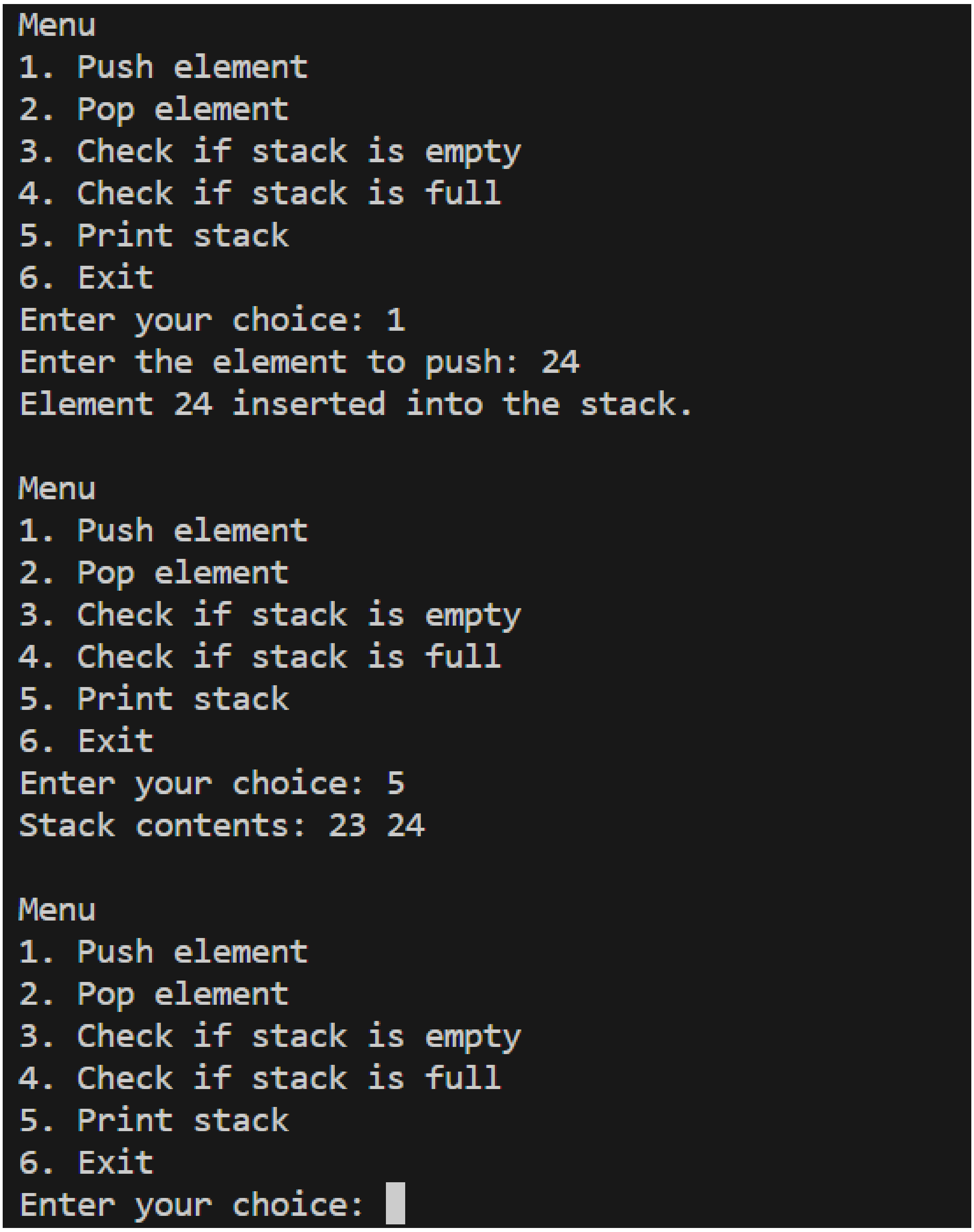
Sample Code:

```
#include <iostream>
using namespace std;
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1; // Top of the stack
```

```
bool isEmpty()
{
return top == -1;
}

bool isFull()
{
return top == MAX_SIZE - 1;
}

void push(int value)
{
if (isFull())
{
cout << "Stack Overflow! Cannot insert element." << endl;return;
}
stack[++top] = value;
cout << "Element " << value << " inserted into the stack." << endl;
}

void pop()
{
if (isEmpty())
{
cout << "Stack Underflow! Cannot delete element." << endl;
#include <iostream>using namespace std;
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1; // Top of the stack


bool isEmpty()
{
return top == -1;
}

bool isFull()
{
return top == MAX_SIZE - 1;
}

void push(int value)
{
if (isFull())
{
cout << "Stack Overflow! Cannot insert element." << endl;return;
}
stack[++top] = value;
cout << "Element " << value << " inserted into the stack." << endl;
}

void pop()
{
if (isEmpty())
{
cout << "Stack Underflow! Cannot delete element." << endl;
break;


case 2:
pop();break;

case 3:
if (isEmpty())
{
cout << "Stack is empty." << endl;
}
else
{
cout << "Stack is not empty." << endl;
}
break;


case 4:
if (isFull())
{
cout << "Stack is full." << endl;
}
else
{
cout << "Stack is not full." << endl;
}
break;

case 5:
printStack();break;

case 6:
cout << "Exiting program.\n";break;

default:
cout << "Invalid choice. Please try again.\n";
}
} while (choice != 6);
```

```
return 0;
}
```

Observations/Outcome:



```
Menu
1. Push element
2. Pop element
3. Check if stack is empty
4. Check if stack is full
5. Print stack
6. Exit
Enter your choice: 1
Enter the element to push: 24
Element 24 inserted into the stack.

Menu
1. Push element
2. Pop element
3. Check if stack is empty
4. Check if stack is full
5. Print stack
6. Exit
Enter your choice: 5
Stack contents: 23 24

Menu
1. Push element
2. Pop element
3. Check if stack is empty
4. Check if stack is full
5. Print stack
6. Exit
Enter your choice:
```

Time Complexity:

Time complexity for push operation is O(1)
Time complexity for pop operation is O(1)
Time complexity for isfull operation is O(1)
Time complexity for isempty operation is O(1)

PRACTICAL FILE

Student Name	Amit Kumar
UID	21BCS9056
Section & Group	21BCS_IOT_644_B
Department	Computer Science & Engineering
Session	July-Dec 2023
Course Name	Design and Analysis of Algorithms with Lab
Course Code	21ITH-311/21CSH-311
Semester	5TH

Department of Computer Science & Engineering
Chandigarh University, Mohali

Course Name: DAA Lab **Course Code: 21ITH-311/21CSH-311**

INDEX

S. No.	Experiment	Date	Conduct (12)	Viva (10)	Worksheet (8)	Total (30)	Remarks
1	**Experiment 1.1:** Analyze if stack Isempty, Isfull and if elements are present then return top element in stacks using templates and also perform push and pop operation in stack.						
2	**Experiment 1.2:** Develop a program for implementation of power function and determine that complexity should be O(log n) .						
3	**Experiment 1.3:** Evaluate the complexity of the developed program to find frequency of elements in a given array.						
4	**Experiment 1.4:** i. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and end of Singly Linked List. ii. Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.						
5	**Experiment 2.1:** Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random						

	number generator.						
6	**Experiment 2.2** Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.						
7	**Experiment 2.3:** Develop a program and analyze complexity to implement 0-1 Knapsack using Dynamic Programming.						
8	**Experiment 3.1:** Develop a program and analyze complexity to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, OR (ii) to find a path from source to goal in a maze						
9	**Experiment 3.2:** Develop a program and analyze complexity to find shortest paths in a graph with positive edge weights using Dijkstra's algorithm.						
10	**Experiment 3.3:** Develop a program and analyze complexity to find all occurrences of a pattern P in a given string S.						

Student Name: Amit Kumar **UID:** 21BCS9056
Branch:BE-CSE **Section/Group:** 644 B
Semester:5 **Date of Performance:**08-08-2023
Subject Name: DAA **Subject Code:** 21CSH-311

Experiment 1.1

Aim: Analyze if stack Isempty, Isfull and if elements are present then return top element in stacksusing templates and also perform push and pop operation in stack.

Procedure/Algorithm:

```
Step 1: Create Stack: Set up a container to store elements.
Step 2: Check if Stack is Empty:
        If there are no elements in the container, the stack is empty.
Step 3: Check if Stack is Full:
        If the number of elements reaches a limit, the stack is full.
Step 4: Push Element onto Stack:
        If the stack is not full:
        Add an element to the container.
        This becomes the new top element.
Step 5: Pop Element from Stack:
        If the stack is not empty:
        Remove the top element from the container.
        The element below it becomes the new top.
Step 6: Get Top Element:
        If the stack is not empty:
        Return the top element without removing it.
```

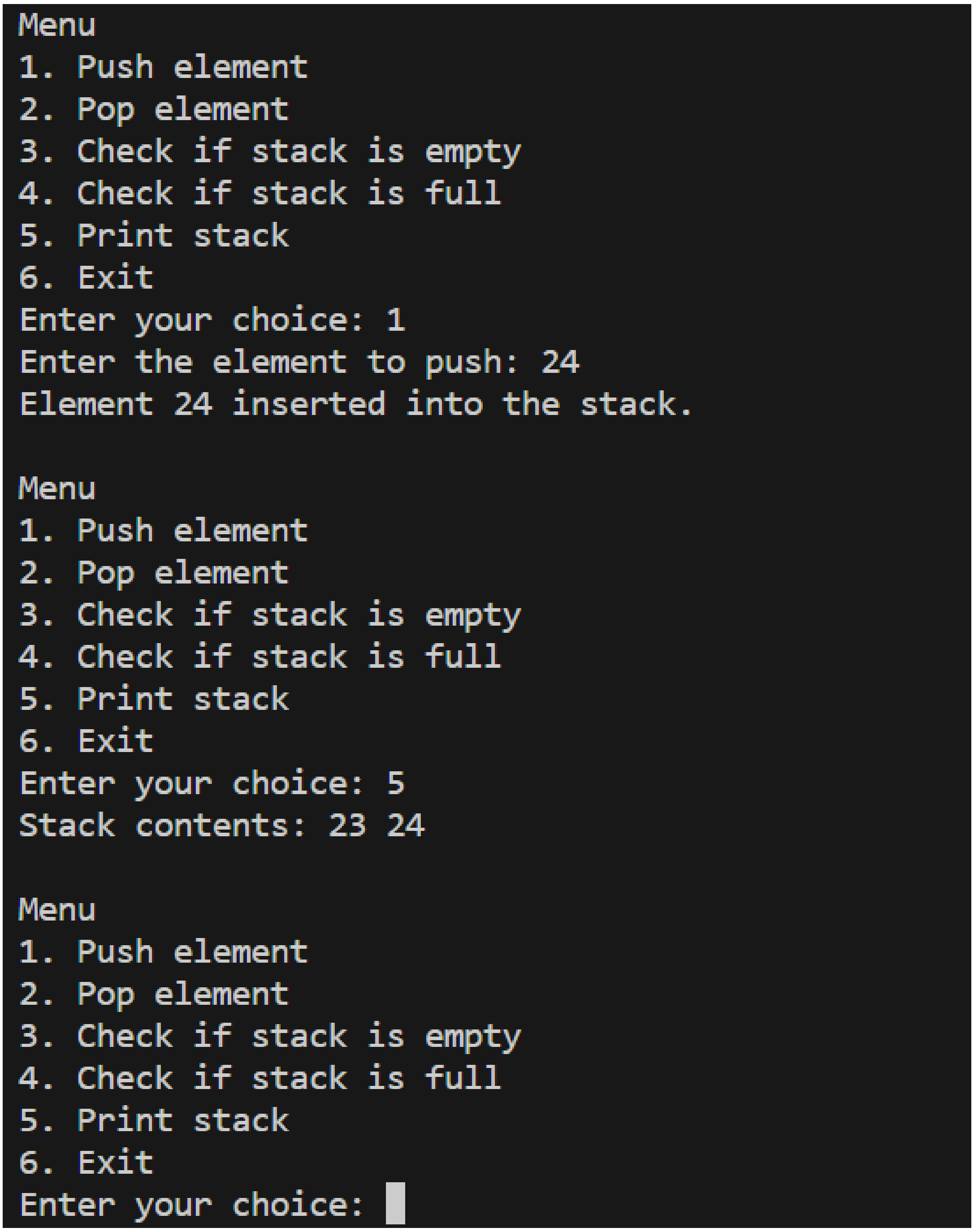
Sample Code:

```
#include <iostream>
using namespace std;
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1; // Top of the stack
```

```
bool isEmpty()
{
return top == -1;
}

bool isFull()
{
return top == MAX_SIZE - 1;
}

void push(int value)
{
if (isFull())
{
cout << "Stack Overflow! Cannot insert element." << endl;return;
}
stack[++top] = value;
cout << "Element " << value << " inserted into the stack." << endl;
}

void pop()
{
if (isEmpty())
{
cout << "Stack Underflow! Cannot delete element." << endl;
#include <iostream>using namespace std;
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1; // Top of the stack


bool isEmpty()
{
return top == -1;
}

bool isFull()
{
return top == MAX_SIZE - 1;
}

void push(int value)
{
if (isFull())
{
cout << "Stack Overflow! Cannot insert element." << endl;return;
}
stack[++top] = value;
cout << "Element " << value << " inserted into the stack." << endl;
}

void pop()
{
if (isEmpty())
{
cout << "Stack Underflow! Cannot delete element." << endl;
break;

case 2:
pop();break;

case 3:
if (isEmpty())
{
cout << "Stack is empty." << endl;
}
else
{
cout << "Stack is not empty." << endl;
}
break;

case 4:
if (isFull())
{
cout << "Stack is full." << endl;
}
else
{
cout << "Stack is not full." << endl;
}
break;

case 5:
printStack();break;

case 6:
cout << "Exiting program.\n";break;

default:
cout << "Invalid choice. Please try again.\n";
}
} while (choice != 6);
```

```
return 0;
}
```

Observations/Outcome:



```
Menu
1. Push element
2. Pop element
3. Check if stack is empty
4. Check if stack is full
5. Print stack
6. Exit
Enter your choice: 1
Enter the element to push: 24
Element 24 inserted into the stack.

Menu
1. Push element
2. Pop element
3. Check if stack is empty
4. Check if stack is full
5. Print stack
6. Exit
Enter your choice: 5
Stack contents: 23 24

Menu
1. Push element
2. Pop element
3. Check if stack is empty
4. Check if stack is full
5. Print stack
6. Exit
Enter your choice: 
```

Time Complexity:

Time complexity for push operation is O(1)
Time complexity for pop operation is O(1)
Time complexity for isfull operation is O(1)
Time complexity for isempty operation is O(1)