SUPERIOR UNIVERSITY

Data Structure and Algorithm (Lab)

Assignment - 1

<u>Name</u>:

Ali Maqsood.

<u>Roll no</u>:

SU92-BSAIM-F23-050.

<u>Department</u>:

Software Engineering Department.

<u>Program</u>:

Artificial Intelligence.

<u>Section</u>:

BSAI-3A

Question # 1:

Objective: Write a program to evaluate a postfix (Reverse Polish Notation) mathematical expression using a stack.

Steps:

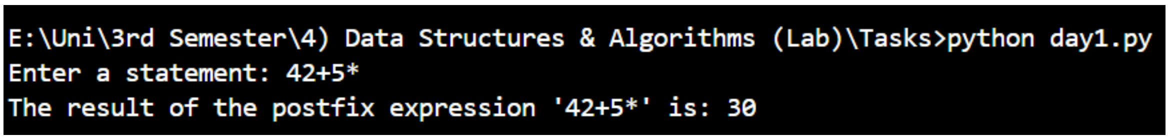
1. Read a postfix expression as input (e.g., "23+5*", which means (2 + 3) * 5).
2. Use a stack to evaluate the expression:
   - Push numbers onto the stack.
   - When encountering an operator, pop two numbers, apply the operator, and push the result back onto the stack.
3. At the end, the stack should contain a single number, which is the result.

Code:

```
class task1:
    def __init__(self):
        self.stack=[]


    def process(self, statement):
        for i in statement:
            if i.isdigit():
                self.stack.append(int(i))
            else:
                a1=self.stack.pop()
                a2=self.stack.pop()
                if i=="+":
                    self.stack.append(a1+a2)
                elif i=="-":
                    self.stack.append(a1-a2)
                elif i=="*":
                    self.stack.append(a1*a2)
                elif i=="/":
```

```
                self.stack.append(a1/a2)
            else:
                print("Invalid operator. Ending....")
                break
        return self.stack.pop()
statement = input("Enter a statement: ")
obj1 = task1()
ans = obj1.process(statement)
print(f"The result of the postfix expression \'{statement}\' is: {ans}")
```

Output:



```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Tasks>python day1.py
Enter a statement: 42+5*
The result of the postfix expression '42+5*' is: 30
```

Question # 2:

Objective: Simulate a simple browser navigation system using two stacks.

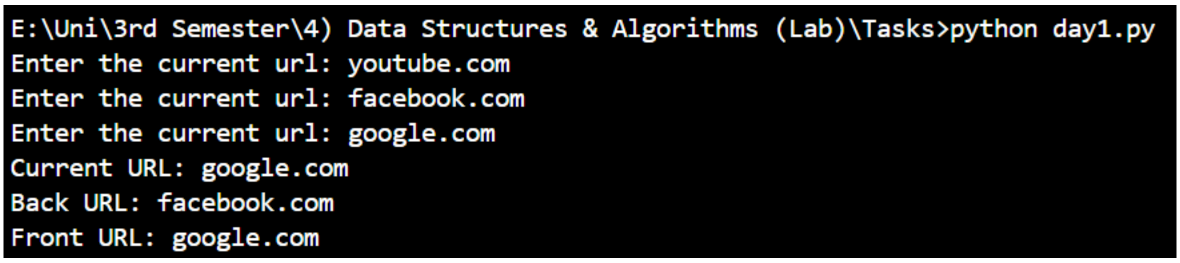
Steps:

1. Create two stacks: back_stack and forward_stack.
2. Implement the following operations:
   - visit(url) - Navigate to a new URL and clear the forward_stack.
   - back() - Move to the previous URL (pop from back_stack and push to forward_stack).
   - forward() - Move to the next URL (pop from forward_stack and push to back_stack).
3. Print the current URL after each operation.

Code:

```
class task2:
    def __init__(self):
        self.back_stack=[]
        self.forward_stack=[]
        self.current_url=None

    def current(self):
        self.current_url=input("Enter the current url: ")
        self.back_stack.append(self.current_url)
        self.forward_stack.clear()

    def previous(self):
        if self.back_stack:
            self.forward_stack.append(self.current_url)
            self.current_url=self.back_stack.pop()
        else:
            print("Back Stack is empty....")

    def forward(self):
        if self.forward_stack:
            self.back_stack.append(self.current_url)
            self.current_url=self.forward_stack.pop()
        else:
            print("Front Stack is empty....")

    def display(self):
        print(f"Current URL: {self.current_url}")
        print(f"Back URL: {self.back_stack.pop()}")
        print(f"Front URL: {self.forward_stack.pop()}")
obj1=task2()
obj1.current()
obj1.current()
obj1.current()
obj1.previous()
obj1.forward()
obj1.display()
```

Output:



```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Tasks>python day1.py
Enter the current url: youtube.com
Enter the current url: facebook.com
Enter the current url: google.com
Current URL: google.com
Back URL: facebook.com
Front URL: google.com
```

Question # 3:

Objective: Write a program to check if a string is a palindrome using a stack.

Steps:

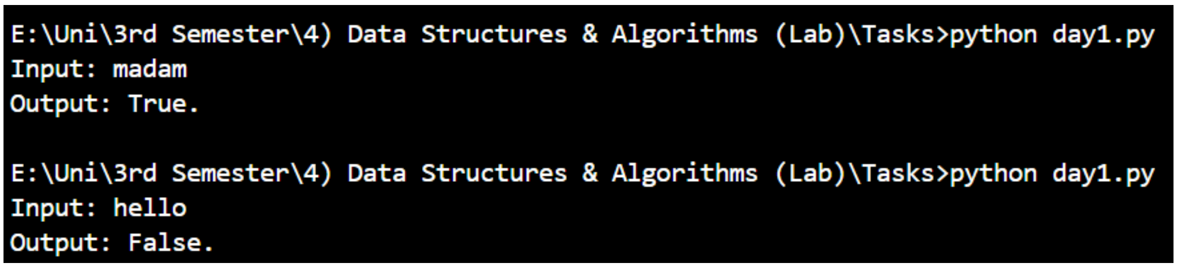
1. Push all characters of the string onto a stack.
2. Pop characters from the stack to create a reversed string.
3. Compare the reversed string with the original string to determine if it is a palindrome.

Code:

```
class task3:
    def __init__(self):
        self.stack = []
        self.input1=input("Input: ").lower()

    def push(self):
        for i in self.input1:
            self.stack.append(i)

    def check(self):
        reversed_word=""
        while self.stack:
            reversed_word+=self.stack.pop()
        if self.input1==reversed_word:
            print("Output: True.")
        else:
            print("Output: False.")
obj1=task3()
obj1.push()
```

```
obj1.check()
```

Output:



```
E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Tasks>python day1.py
Input: madam
Output: True.

E:\Uni\3rd Semester\4) Data Structures & Algorithms (Lab)\Tasks>python day1.py
Input: hello
Output: False.
```