**GIT Department of Computer Engineering**

**CSE 222/505 - Spring 2020**

**Homework 4 Report**

**Muhammed Eyüp**

**1801042679**

**Q1**

1. **Class Diagram:**

No class diagram because no code was written

1. **Problem Solution Approach:**

In the process of converting an expression from infix to both prefix and postfix forms Stack was used.

When converting to prefix the expression was reversed before converting and scanned from left to write then reversed back after the conversion was over to get the final result.

When converting to postfix the expression was scanned from left to write as is.

When Evaluating the prefix expression, it was scanned from right to left to get the result.

When Evaluating the postfix expression, it was scanned from left to right to get the result.

1. **Test cases:**

No test cases because no code was written

1. **Running Commands and Results:**

No Running Commands and Results because no code was written

**Q2**

1. **Class Diagram:**

**A close up of text on a white background

Description automatically generated**

1. **Problem Solution Approach:**

In the Deque the problem was to make a queue that can be accessed from the beginning and the end in other words we can add elements to its head and tail the problem was solved by using a Doubly Linked List which has a head and tail so it can be accessed from both the beginning and the end and another problem was the need to create a new node after deleting and re-adding an element and this problem was solved by having another linked list that holds on to the nodes that has been removed from the main deque and then reusing these node as new nodes for the deque whenever a new element has been added

1. **Test cases:**

|  |  |  |
| --- | --- | --- |
| Scenario | Test Case | Result |
| Adding to the head | Element provided | Adds to the head |
| Adding to the tail | Element provided | Adds to the tail |
| Peeking from the head | Deque is not empty | Returns the head element |
| Deque is empty (get function is used) | Throws a NoSuchElementException |
| Deque is empty (peek function is used) | Returns null |
| Peeking from the tail | Deque is not empty | Returns the tail element |
| Deque is empty (get function is used) | Throws a NoSuchElementException |
| Deque is empty (peek function is used) | Returns null |
| Remove from the head | Deque is not empty | Removes and returns the head element |
| Deque is empty (remove function is used) | Throws a NoSuchElementException |
| Deque is empty (poll function is used) | Returns null |
| Remove from the tail | Deque is not empty | Removes and returns the tail element |
| Deque is empty (remove function is used) | Throws a NoSuchElementException |
| Deque is empty (poll function is used) | Returns null |

1. **Running Commands and Results:**

A picture containing table, laptop, keyboard

Description automatically generated

**Q3**

1. **Class Diagram:**

A close up of text on a white background

Description automatically generated

1. **Problem Solution Approach:**

For the isElfish problem the problem was to determine if a word as the letters e, l and f for this problem I set a counter so the base case was when the string is over and the counter was less than 3 we stop the function and return false otherwise if the counter reached 3 there is no need to keep going we just return true.

The smaller problem was to check each character if it was any of the three if it was, we increment the counter and call the function with the next character.

For the selection sort problem, the base case is when we reach the end of the array, other than that we use a very simple Selection sort algorithm.

For the prefix problem we start from the end of the string and we use a stack to push the operands to when we encounter an operator we pop two elements from the stack and evaluate them then we push the result to the stack and the base case is when we reach the beginning of the string

For the postfix problem we start from the beginning of the string and we use a stack to push the operands to. when we encounter an operator, we pop two elements from the stack and evaluate them then we push the result to the stack and the base case is when we reach the end of the string

1. **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Scenario | Test Case | Result |
| isElfish Function | Provide an elfish word | Returns true |
| Provide a non-elfish word | Returns false |
| Selection sort | Provide an array of numbers | Sorts the array |
| Prefix evaluation | Provide a prefix expression | Evaluates and returns the result |
| Postfix evaluation | Provide a postfix expression | Evaluates and returns the result |

1. **Running Commands and Results:**

**A picture containing black, keyboard

Description automatically generated**