GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework #4 Part 3 Report

Murat YILDIZ
1801042004

PROBLEM SOLUTION APPROACH

Define each of the following problem recursively:

- Identify the base case (or base cases) that stops the recursion
- Define the smaller problem (or problems)
- Explain how to combine solutions of smaller problems to get the solution of original problem

1. reverseString(String str)

Base case: if there is no ‘ ‘ in string, it stops

Smaller problem: take a string (named splitStr) until encounter with ‘ ‘ , after that call itself with a string begins index of ‘ ‘ +1 and ends with str.length

Combine: return reverseString(`str.substring(str.indexOf(" ") + 1 ) + “ “ + splitStr)`

2. isElfish(String word) uses isElfishHelper(String word, String isElfish)

Base cases: isElfish string has 3 elements in it or if there is no character to read in word

Smaller problem: read first char of string and control it if it is ‘e’ ‘l’ ‘f’ put it to isElfish(if not there), then call it self smalller string with *isElfishHelper*(word.substring(1), isElfish)

Combine: return *isElfishHelper*(word.substring(1), isElfish)

3. selectionSort(int arr[]) uses selectionSortHelper(int arr[], int startIndex) uses findMinValueIndex(int arr[], int startIndex)

Base cases: if startIndex is lower than arr.length

Smaller problem: find min value index from startIndex to arr.length (findMinValueIndex) and swap min value index and startIndex

Combine: *selectionSortHelper*(arr, startIndex+1);

4. evaluatePrefix(String expression) uses evaluatePrefixHelper(String expression, Stack <Double> stack)

Base cases: if there is no ‘ ‘ in string, it stops and returns stack.pop()

Smaller problem: read backward from the end of string take a string until encounter with ‘ ‘ (index of ‘ ‘ is cursor) , and control it wheter it is operator or not, do some operation according to it , then call itself with smaller string expression.substring(0,cursor)

Combine: return *evaluatePostfixHelper*(expression.substring(0,cursor)stack)

5. evaluatePostfix(String expression) uses evaluatePostfixHelper(String expression, Stack <Double> stack)

Base cases: if there is no ‘ ‘ in string, it stops and returns stack.pop()

Smaller problem: read forward take a string until encounter with ‘ ‘ (index of ‘ ‘ is cursor) , and control it wheter it is operator or not, do some operation according to it , then call itself with smaller string expression.substring(cursor+1)

Combine: return *evaluatePostfixHelper*(expression.substring(cursor+1), stack)

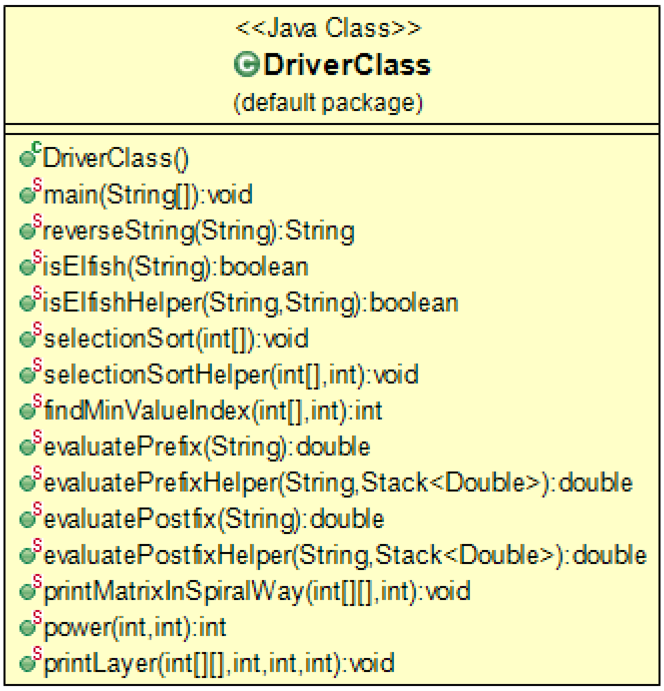
6. printMatrixInSpiralWay(int [][] arr, int count) uses printLayer(int [][] arr, int rowPosition, int colPosition, int count)

Base cases: if small edge's value < 2 ^ count +1 (count is which layer to be printed)

Smaller problem: call printLayer and print it, then call itself to go second layer of matrix

Combine: return *printMatrixInSpiralWay*(arr, ++count) ,

CLASS DIAGRAM


<<Java Class>>
DriverClass
(default package)

DriverClass()
main(String[]):void
reverseString(String):String
isElfish(String):boolean
isElfishHelper(String,String):boolean
selectionSort(int[]):void
selectionSortHelper(int[],int):void
findMinValueIndex(int[],int):int
evaluatePrefix(String):double
evaluatePrefixHelper(String,Stack<Double>):double
evaluatePostfix(String):double
evaluatePostfixHelper(String,Stack<Double>):double
printMatrixInSpiralWay(int[][],int):void
power(int,int):int
printLayer(int[][],int,int,int):void

TEST CASES

TEST ID	Scenario	Test Data	Expected Results	Actual Results	Pass/ Fail
TEST01	reverseString(str) , when str is null		Exception Throwed	As Expected	PASS
TEST02	reverseString(str)	str = homework 222 CSE for test a is This	Successfully Done	As Expected	PASS
TEST03	isElfish(str) , when str is null		Exception Throwed	As Expected	PASS
TEST04	isElfish(str)	str = whiteleaf , Returned: true	Successfully Done	As Expected	PASS
TEST05	isElfish(str)	str = whiteeeaf , Returned: false	Successfully Done	As Expected	PASS
TEST06	selectionSort(arr)	arr = 3, 7, 0, 6, 13, 3, 1, 8, 19, 4,	Successfully Done	As Expected	PASS
TEST07	evaluatePostfix(strPostfix)	strPostfix = 20 15 2 6 * - 3 / + 8 + 16 4 / - Result: 25,000000	Successfully Done	As Expected	PASS
TEST08	evaluatePrefix(strPrefix)	strPrefix = + 20 + / - 15 * 2 6 3 - 8 / 16 4 Result: 25,000000	Successfully Done	As Expected	PASS
TEST09	evaluatePostfix(strPostfix) , when strPostfix is null		Exception Throwed	As Expected	PASS
TEST10	evaluatePrefix(strPrefix) , when strPrefix is null		Exception Throwed	As Expected	PASS
TEST11	printMatrixInSpiralWay(arr1, 0)		Successfully Done	As Expected	PASS
TEST12	printMatrixInSpiralWay(arr2, 0)		Successfully Done	As Expected	PASS
TEST13	printMatrixInSpiralWay(arr3, 0)		Successfully Done	As Expected	PASS
TEST14	printMatrixInSpiralWay(arr4, 0)		Successfully Done	As Expected	PASS
TEST15	printMatrixInSpiralWay(arr5, 0)		Successfully Done	As Expected	PASS
TEST16	printMatrixInSpiralWay(arr6, 0)		Successfully Done	As Expected	PASS

RUNNING AND RESULTS

```
 TEST01 - reverseString(str) , when str is null

java.lang.NullPointerException
```

```
 TEST02 - reverseString(str)

homework 222 CSE for test a is This
```

```
 TEST03 - isElfish(str) , when str is null

java.lang.NullPointerException
```

```
 TEST04 - isElfish(str), str = whiteleaf , Returned: true
```

```
 TEST05 - isElfish(str), str = whiteeeaf , Returned: false
```

```
 TEST06 - selectionSort(arr),
 arr = 3, 7, 0, 6, 13, 3, 1, 8, 19, 4,
 After selectionSort(arr),
 arr = 0, 1, 3, 3, 4, 6, 7, 8, 13, 19,
```

```
TEST07 - evaluatePostfix(strPostfix), strPostfix = 20 15 2 6 * - 3 / + 8 + 16 4 / -
Result: 25,000000
```

```
TEST08 - evaluatePrefix(strPrefix), strPrefix = + 20 + / - 15 * 2 6 3 - 8 / 16 4
Result: 25,000000
```

```
 TEST09 - evaluatePostfix(strPostfix) , when strPostfix is null

java.lang.NullPointerException
```

```
 TEST10 - evaluatePrefix(strPrefix) , when strPrefix is null

java.lang.NullPointerException
```

```
int [][]arr1 = { { 1, 2, 3, 4},
                 { 5, 6, 7, 8},
                 { 9,10,11,12},
                 {13,14,15,16},
                 {17,18,19,20}   };
```

```
TEST11 - printMatrixInSpiralWay(arr1, 0)
1, 2, 3, 4, 8, 12, 16, 20, 19, 18, 17, 13, 9, 5, 6, 7, 11, 15, 14, 10,
```

```
int [][]arr2 = {{1,2,3}};
```

```
TEST12 - printMatrixInSpiralWay(arr2, 0)
1, 2, 3,
```

```
int [][]arr3 = {{1},{2},{3}};
```

```
TEST13 - printMatrixInSpiralWay(arr3, 0)
1, 2, 3,
```

```
int [][]arr4 = {{1,2,3}, {3,4,5}};
```

```
TEST14 - printMatrixInSpiralWay(arr4, 0)
1, 2, 3, 5, 4, 3,
```

```
int [][]arr5 = {{1,2},{3,4},{5,6}};
```

```
TEST15 - printMatrixInSpiralWay(arr5, 0)
1, 2, 4, 6, 5, 3,
```

```
int [][]arr6 = {
                {1,2,3,4,5},
                {6,7,8,9,10},
                {11,12,13,14,15},
                {16,17,18,19,20},
                {21,22,23,24,25}    };
```

```
TEST16 - printMatrixInSpiralWay(arr6, 0)
1, 2, 3, 4, 5, 10, 15, 20, 25, 24, 23, 22, 21, 16, 11, 6, 7, 8, 9, 14, 19, 18, 17, 12, 13,
```