GIT Department of Computer Engineering

CSE 222/505 - Spring 2020

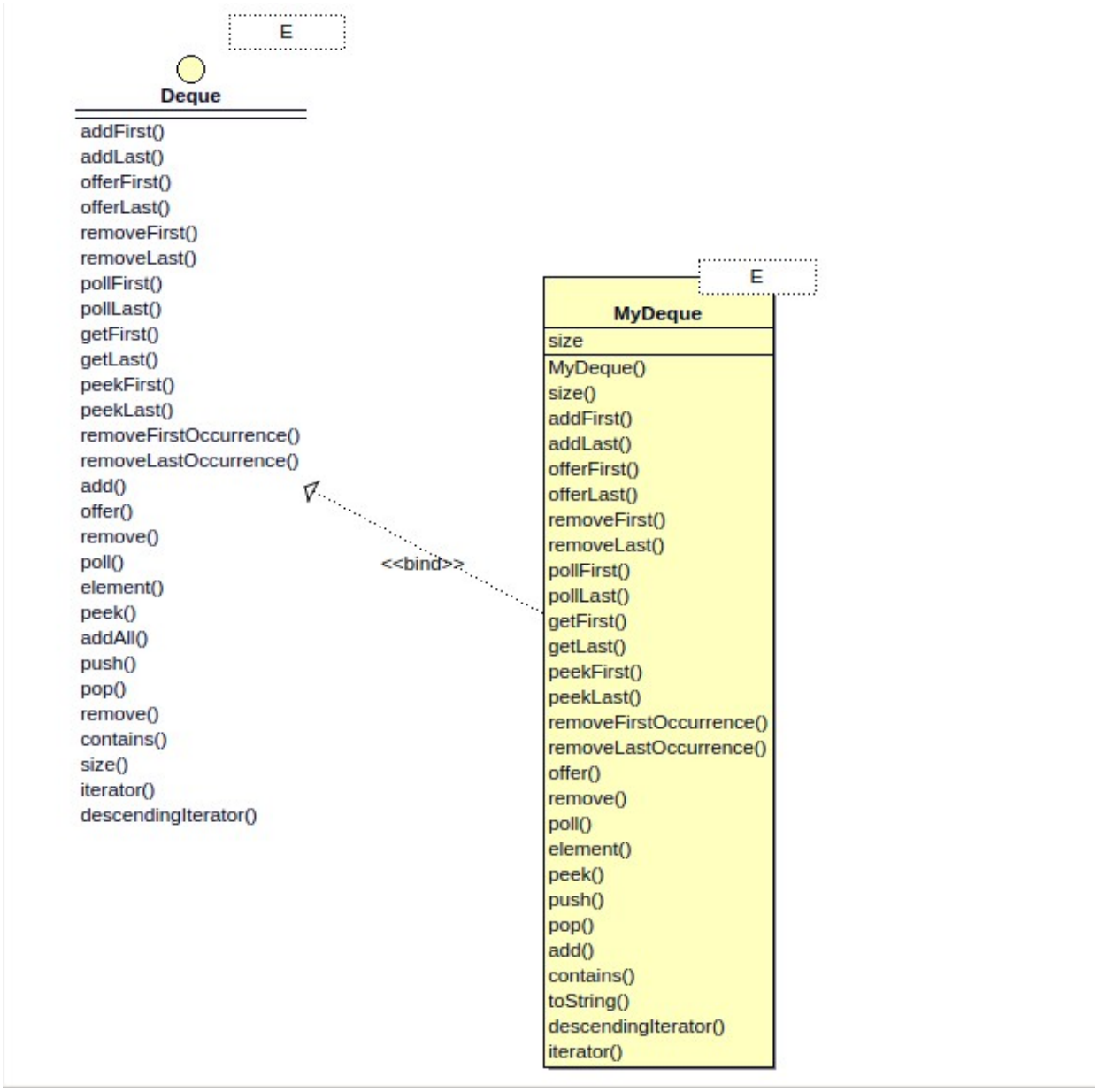
Homework 4 Report

Muharrem Ozan Yeşiller

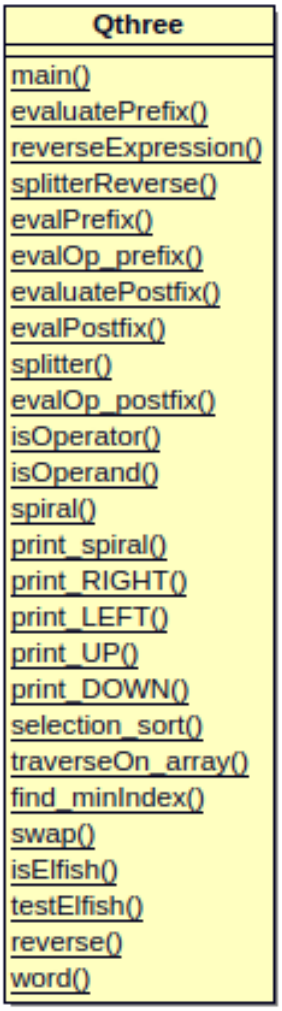
171044033

1. CLASS DIAGRAMS

1.1(Question2)



1.2(Question3)



2. PROBLEM SOLUTION APPROACH

2.1(Question2)

When implementing the deque data structure, what we need to pay attention to is access from the front and rear because deque expansion is double ended queued. So these accesses cannot be used in our external data structure but also throws an exception. After we designed our problem solution accordingly, we kept a double linkedlist structure inside, since our operations were constant time (O(1)). Our data structure provides basic data structure features such as adding, removing, searching etc. I implemented these features with the help of iterator. And at the same time, I used the memory more efficiently by collecting garbage by holding the removed nodes together. While meeting these needs, i implemented the deque interface and extended from the AbstractCollection class. And now i just had to implement the methods we needed, I implemented these methods and solved the problem.

2.2(Question3)

2.2.1

In this section, print reverse the sentence word by word. Not letter by letter! This is a smaller problem. So the problem has to be divided into pieces. The first part of the problem is taking words. I have solved this problem by implementing a method that divides the sentence word by word and by printing the feedback of this method in reverse. To separate the sentence word by word, I examined the character character and if there is a whitespace or end of line, I appended the word from the index I started to the last index to StringBuilder. This is a smaller problem. And then I printed these words starting from the end. While doing these operations, the base case is the size of the main sentence as a character.

2.2.2

In this section, the word requested from us includes the letters e, f and l. The order of the letters in this is insignificant. Therefore, when I read the letter character by character and reach the required letter, I assign the counter of that letter is TRUE. This is a smaller problem. When we come to the end of the word as the best case. If the counters for e, l and f are TRUE, the word is elfish. Base case is length of input or (e count and l count and f count are true.

2.2.3

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list. The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right. Our parts here are sorted and unsorted array. Our problem is to perform these operations in a recursive way as much as the size of the base case.

2.2.4

First of all, I divided the tokens in each expression one by one. This is a smaller problem. Then we need to evaluate the token we taken. These parts can be operator or operand. If they are not, then we are processing a non prefix expression. If the token we received is an operand, the operand is pushed to stack. If the token we received is an operator, the stack is processed with 2 pops and pushed to the stack. First pop is right value. The only element left in the last stack is the solution. If there are more than 1 element in the stack, the input is non prefix expression.

How do we make these operations according to a statement? We do this by reading the prefix statement in reverse.

Method continues length of expression recursively(this is base case).

2.2.5

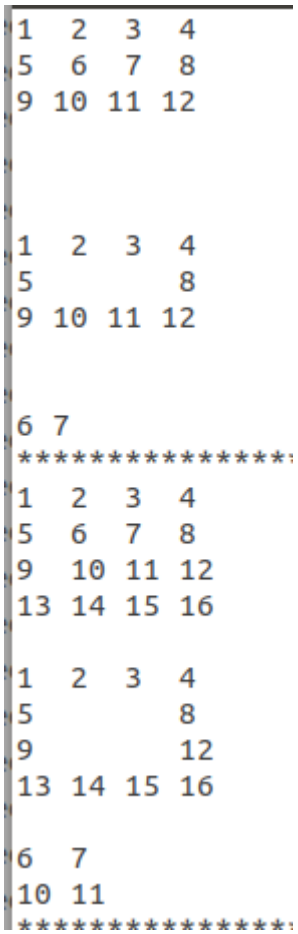
First of all, I divided the tokens in each expression one by one. This is a smaller problem. Then we need to evaluate the token we taken. These parts can be operator or operand. If they are not, then we are processing a non postfix expression. If the token we received is an operand, the operand is pushed to stack. If the token we received is an operator, the stack is processed with 2 pops and pushed to the stack. First pop is left value. The only element left in the last stack is the solution. If there are more than 1 element in the stack, the input is non postfix expression.

How do we make these operations according to a statement? We do this by reading the prefix statement in not reverse, reading from beginning of expression.

Method continues length of expression recursively(this is base case).

2.2.6

First we need to cut the perimeter of the matrix. This is a smaller problem of this part. We need to prints the matrix, which we divide from its surroundings, to the right, then to the left, then down and then up. There is a smaller problem here to print right, left, up and down.



The number of columns is our base case, moving to the right. As the number of rows moves down. There is a problem here when we go left and up. we lose the indexes, as these processes are performed by different methods. Therefore, when moving to the left, we reverse print from our coordinates in the bottom left corner. The same is true in Up. We reverse print down from the top left corner. The base case is element of array. When we print as much as the total element of the array, the method completes its task.

3. TEST CASES

Tests were made explicitly in the main program. The main purposes in these tests are to try out endpoints. to get the right results in accordance with the solution of the problem.

Question 2

```
System.out.println("Add First Test");
System.out.println("Adding some words");
test.addFirst("Ozan");
test.addFirst("171044033");
test.add("Yesiller");
System.out.print(test);
System.out.println("Add first test success");

System.out.println("\nAdd Last Test");
System.out.println("Adding some words");
test.addLast("Gebze");
test.addLast("Technical");
test.addLast("University");
System.out.print(test);
System.out.println("Add Last test success");

System.out.println("\nOffer first Test");
System.out.println("Adding some words");
test.offerFirst("Basak");
test.offerFirst("Karakas");
test.offerFirst("GTU");
System.out.print(test);
System.out.println("Offer first test success");

System.out.println("\nOffer last Test");
System.out.println("Adding some words");
test.offerLast("Fatih");
test.offerLast("Erdogan");
test.offerLast("Sevilgen");
System.out.print(test);
System.out.println("Offer last test success");
```

```
System.out.println("\nRemove first test");
test.removeFirst();
System.out.println(test);
test.removeFirst();
System.out.println(test);
test.removeFirst();
System.out.println(test);
System.out.println("Remove first test success");

System.out.println("\nRemove last test");
test.removeLast();
System.out.println(test);
test.removeLast();
System.out.println(test);
test.removeLast();
System.out.println(test);
System.out.println("Remove last test success");

System.out.println("\nPoll first test");
test.pollFirst();
System.out.println(test);
test.pollFirst();
System.out.println(test);
test.pollFirst();
System.out.println(test);
System.out.println("Poll first test success");

System.out.println("\nPoll last test");
test.pollLast();
System.out.println(test);
test.pollLast();
System.out.println(test);
test.pollLast();
System.out.println(test);
System.out.println("Poll last test success");
```

```java
System.out.println("\nadd(E e) and garbage collecting test");
test.add("GTU");
test.add("CSE222");
test.add("ALGO&DAT");
System.out.println(test);

System.out.println("Contains Test");
System.out.println("Searching 'CSE222'");
if(test.contains("CSE222"))
    System.out.println("Found");
else
    System.out.println("Not found");

System.out.println("Searching 'Ozan'");
if(test.contains("Ozan"))
    System.out.println("Found");
else
    System.out.println("Not found");


System.out.println("\nPush(E e) test");
test.push("Nur");
System.out.println(test);
test.push("Banu");
System.out.println(test);
test.push("Albayrak");
System.out.println(test);
System.out.println("Test success");


System.out.println("\nPop test");
test.pop();
System.out.println(test);
test.pop();
System.out.println(test);
test.pop();
System.out.println(test);
System.out.println("Test success");
```

```java
System.out.println("\nGet test");
System.out.println("The getFirst() is -> " + test.getFirst());
System.out.println("The getLast() is ->  " + test.getLast());
System.out.println("The element() is -> " + test.element());
System.out.println("Test success");

System.out.println("\nPeek Test");
System.out.println("The peekFirst() is -> " + test.peekFirst());
System.out.println("The peekLast() is -> " + test.peekLast());
System.out.println("The peek() is -> " + test.peek());
System.out.println("Test success");

System.out.println("\nOffer() test");
test.offer("GTU");
test.offer("Faculty of");
test.offer("Computer Engineering");
System.out.println(test);
System.out.println("Test success");

System.out.println("\nRemove First Occurence of GTU");
test.removeFirstOccurrence("GTU");
System.out.println(test);
System.out.println("Test success");

System.out.println("\nRemove Last Occurence of GTU");
test.removeLastOccurrence("GTU");
System.out.println(test);
System.out.println("Test success");

System.out.println("\npoll() test(two times)");
test.poll();
test.poll();
System.out.println(test);
System.out.println("Test success");

System.out.println("\nremove() test(two times)");
test.remove();
test.remove();
System.out.println(test);
System.out.println("Test Success");
```

Question 3

-Part1

```
System.out.println("QUESTION 3 PART 1 TEST: \n");
System.out.println("1) " + "this function writes the sentence in reverse");
reverse("this function writes the sentence in reverse", 0);
System.out.println("\n2) " + "CSE222 Data Structures And Algorithms");
reverse("CSE222 Data Structures And Algorithms", 0);
System.out.println("\n3) " + "this interface does not provide support for indexed access to elements");
reverse("this interface does not provide support for indexed access to elements", 0);
System.out.println("\nEnd of the test\n");
```

-Part2

```
System.out.println("QUESTION 3 PART 2 TEST: \n");

System.out.print("1) elfish");
if(isElfish("elfish"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");


System.out.print("2) pencil");
if(isElfish("pencil"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");


System.out.print("3) whiteleaf");
if(isElfish("whiteleaf"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");


System.out.print("4) freq");
if(isElfish("freq"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");
```

```
System.out.print("5) tasteful");
if(isElfish("tasteful"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");


System.out.print("6) rampage");
if(isElfish("rampage"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");


System.out.print("7) unfriendly");
if(isElfish("elfish"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");


System.out.print("8) waffles");
if(isElfish("waffles"))
    System.out.println(" is 'elfish'...");
else
    System.out.println(" is not 'elfish'...");

System.out.println("\nEnd of the test\n");
```

-Part3

```
System.out.println("QUESTION 3 PART 3 TEST: \n");

int[] unsorted = { 64, 25, 12, 22, 11 };
System.out.println("1)The array is: ");
for (int value : unsorted) System.out.print(value + " ");
selection_sort(unsorted);
System.out.println("Sorted...\nResult: ");
for (int value : unsorted) System.out.print(value + " ");
System.out.println("");


int[] unsorted2 = { 12, 11, 13, 5, 6 };
System.out.println("\n2)The array is: ");
for (int value : unsorted2) System.out.print(value + " ");
selection_sort(unsorted2);
System.out.println("Sorted...\nResult: ");
for (int value : unsorted2) System.out.print(value + " ");
System.out.println("");


int[] unsorted3 = { -5, -10, 0, -3, 8, 5, -1, 10 };
System.out.println("\n3)The array is: ");
for (int value : unsorted3) System.out.print(value + " ");
selection_sort(unsorted3);
System.out.println("Sorted...\nResult: ");
for (int value : unsorted3) System.out.print(value + " ");
System.out.println("");


int[] unsorted4 = { 40, 60, 1, 200, 9, 83, 17 ,-5, -10, 0, -3, 8, 5, -1, 10 };
System.out.println("\n4)The array is: ");
for (int value : unsorted4) System.out.print(value + " ");
selection_sort(unsorted4);
System.out.println("Sorted...\nResult: ");
for (int value : unsorted4) System.out.print(value + " ");
System.out.println("");

System.out.println("\nEnd of the test\n");
```

-Part4 and 5

```
System.out.println("QUESTION 3 PART 4 AND 5 TEST: \n");

System.out.println("\nPostfix expression: " + "10 50 5 2 * - 10 / + 2 + 21 7 / -");
System.out.println("Result: " + evaluatePostfix("10 50 5 2 * - 10 / + 2 + 21 7 / -"));

System.out.println("\nPostfix expression: " + "8 2 * 3 * 3 - 8 4 / 1 1 + / +");
System.out.println("Result: " + evaluatePostfix("8 2 * 3 * 3 - 8 4 / 1 1 + / +"));

System.out.println("\nPostfix expression: " + "2 3 1 * + 9 -");
System.out.println("Result: " + evaluatePostfix("2 3 1 * + 9 -"));

try{
    System.out.println("\nPostfix expression: " + "12 2 + -");
    System.out.println("Result: " + evaluatePostfix("12 2 + -"));
}catch (Exception e){
    System.out.println(e.getMessage());
    System.out.println("This is a not postfix expression!!!");
}

System.out.println("\nPrefix expression: " + "- + * 2 3 * 5 4 9");
System.out.println("Result: " + evaluatePrefix("- + * 2 3 * 5 4 9"));

System.out.println("\nPrefix expression: " + "- + + 10 / - 50 * 5 2 10 2 / 21 7");
System.out.println("Result: " + evaluatePrefix("- + + 10 / - 50 * 5 2 10 2 / 21 7"));

System.out.println("\nPrefix expression: " + "+ - * 2 2 / 16 8 5");
System.out.println("Result: " + evaluatePrefix("+ - * 2 2 / 16 8 5"));

try{
    System.out.println("\nPrefix expression: " + "- 10 2 +");
    System.out.println("Result: " + evaluatePrefix("- 10 2 +"));
}catch (Exception e){
    System.out.println(e.getMessage());
    System.out.println("This is a not prefix expression!!!");
}

System.out.println("\nEnd of the test\n");
```

-Part6

```
System.out.println("QUESTION 3 PART 6 TEST: \n");

int[][] ar = {
        {1,2,3,4},
        {5,6,7,8},
        {9,10,11,12},
        {13,14,15,16}
};

for(int i = 0 ; i < ar.length ; ++i)
{
    for(int j = 0 ; j < ar[i].length ; ++j) {
        if(ar[i][j] >= 10 || ar[i][j] <= -10)
            System.out.print(ar[i][j] + " ");
        else
            System.out.print(ar[i][j] + "  ");
    }
    System.out.println("");
}

System.out.print("\nSpiral of this array: ");
spiral(ar);
System.out.println("\n");
```

```
int[][] ar2 = {
        {1,2,3,4},
        {5,6,7,8},
        {9,10,11,12},
        {13,14,15,16},
        {17,18,19,20}
};

for(int i = 0 ; i < ar2.length ; ++i)
{
    for(int j = 0 ; j < ar2[i].length ; ++j) {
        if(ar2[i][j] >= 10 || ar2[i][j] <= -10)
            System.out.print(ar2[i][j] + " ");
        else
            System.out.print(ar2[i][j] + "  ");
    }
    System.out.println("");
}

System.out.print("\nSpiral of this array: ");
spiral(ar2);
System.out.println("\n");

 int ar3[][] = { { 1, 2, 3, 4, 5, 6 },
         { 7, 8, 9, 10, 11, 12 },
         { 13, 14, 15, 16, 17, 18 } };

 for(int i = 0 ; i < ar3.length ; ++i)
 {
     for(int j = 0 ; j < ar3[i].length ; ++j) {
         if(ar3[i][j] >= 10 || ar3[i][j] <= -10)
             System.out.print(ar3[i][j] + " ");
         else
             System.out.print(ar3[i][j] + "  ");
     }

     System.out.println("");
 }

 System.out.print("\nSpiral of this array: ");
 spiral(ar2);
 System.out.println("\n");

 System.out.println("\nEnd of the test");
 System.out.println("**************************
 ////////////////////////////////////////////
```

4. RUNNING COMANDS AND RESULTS

Question 2

```
Add First Test
Adding some words
GARBAGE OF DEQUE:

--------------
Content of Deque:
171044033, Ozan, Yesiller
Add first test success

Add Last Test
Adding some words
GARBAGE OF DEQUE:

--------------
Content of Deque:
171044033, Ozan, Yesiller, Gebze, Technical, University
Add Last test success
```

```
Offer first Test
Adding some words
GARBAGE OF DEQUE:

--------------
Content of Deque:
GTU, Karakas, Basak, 171044033, Ozan, Yesiller, Gebze, Technical, University
Offer first test success

Offer last Test
Adding some words
GARBAGE OF DEQUE:

-------------
Content of Deque:
GTU, Karakas, Basak, 171044033, Ozan, Yesiller, Gebze, Technical, University, Fatih, Erdogan, Sevilgen
Offer last test success

Remove first test
GARBAGE OF DEQUE:
GTU
-------------
Content of Deque:
Karakas, Basak, 171044033, Ozan, Yesiller, Gebze, Technical, University, Fatih, Erdogan, Sevilgen

GARBAGE OF DEQUE:
Karakas, GTU
-------------
Content of Deque:
Basak, 171044033, Ozan, Yesiller, Gebze, Technical, University, Fatih, Erdogan, Sevilgen

GARBAGE OF DEQUE:
Basak, Karakas, GTU
-------------
Content of Deque:
171044033, Ozan, Yesiller, Gebze, Technical, University, Fatih, Erdogan, Sevilgen
```

```
Remove last test
GARBAGE OF DEQUE:
Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
171044033, Ozan, Yesiller, Gebze, Technical, University, Fatih, Erdogan

GARBAGE OF DEQUE:
Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
171044033, Ozan, Yesiller, Gebze, Technical, University, Fatih

GARBAGE OF DEQUE:
Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
171044033, Ozan, Yesiller, Gebze, Technical, University

Remove last test success


Poll first test
GARBAGE OF DEQUE:
171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
Ozan, Yesiller, Gebze, Technical, University

GARBAGE OF DEQUE:
Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
Yesiller, Gebze, Technical, University

GARBAGE OF DEQUE:
Yesiller, Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
Gebze, Technical, University

Poll first test success


Poll last test
GARBAGE OF DEQUE:
University, Yesiller, Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
Gebze, Technical

GARBAGE OF DEQUE:
Technical, University, Yesiller, Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
Gebze

GARBAGE OF DEQUE:
Gebze, Technical, University, Yesiller, Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:


Poll last test success
```

```
add(E e) and garbage collecting test
GARBAGE OF DEQUE:
Yesiller, Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
GTU, CSE222, ALGO&DAT

Contains Test
Searching 'CSE222'
Found
Searching 'Ozan'
Not found

Push(E e) test
GARBAGE OF DEQUE:
Ozan, 171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
Nur, GTU, CSE222, ALGO&DAT

GARBAGE OF DEQUE:
171044033, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
Banu, Nur, GTU, CSE222, ALGO&DAT

GARBAGE OF DEQUE:
Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
Albayrak, Banu, Nur, GTU, CSE222, ALGO&DAT

Test success

Pop test
GARBAGE OF DEQUE:
Albayrak, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
Banu, Nur, GTU, CSE222, ALGO&DAT

GARBAGE OF DEQUE:
Banu, Albayrak, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
Nur, GTU, CSE222, ALGO&DAT

GARBAGE OF DEQUE:
Nur, Banu, Albayrak, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
-------------
Content of Deque:
GTU, CSE222, ALGO&DAT

Test success
```

```
Get test
The getFirst() is -> GTU
The getLast() is ->  ALGO&DAT
The element() is -> GTU
Test success

Peek Test
The peekFirst() is -> GTU
The peekLast() is -> ALGO&DAT
The peek() is -> GTU
Test success

Offer() test
GARBAGE OF DEQUE:
Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
GTU, CSE222, ALGO&DAT, GTU, Faculty of, Computer Engineering

Test success

Remove First Occurence of GTU
GARBAGE OF DEQUE:
GTU, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
CSE222, ALGO&DAT, GTU, Faculty of, Computer Engineering

Test success
```

```
Remove Last Occurence of GTU
GARBAGE OF DEQUE:
GTU, GTU, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
CSE222, ALGO&DAT, Faculty of, Computer Engineering

Test success

poll() test(two times)
GARBAGE OF DEQUE:
ALGO&DAT, CSE222, GTU, GTU, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:
Faculty of, Computer Engineering

Test success

remove() test(two times)
GARBAGE OF DEQUE:
Computer Engineering, Faculty of, ALGO&DAT, CSE222, GTU, GTU, Fatih, Erdogan, Sevilgen, Basak, Karakas, GTU
--------------
Content of Deque:


Test Success
```

Question3

```
**************************************************
QUESTION 3 PART 1 TEST:

1) this function writes the sentence in reverse
reverse in sentence the writes function this

2) CSE222 Data Structures And Algorithms
Algorithms And Structures Data CSE222

3) this interface does not provide support for indexed access to elements
elements to access indexed for support provide not does interface this

End of the test

**************************************************
QUESTION 3 PART 2 TEST:

1) elfish is 'elfish'...
2) pencil is not 'elfish'...
3) whiteleaf is 'elfish'...
4) freq is not 'elfish'...
5) tasteful is 'elfish'...
6) rampage is not 'elfish'...
7) unfriendly is 'elfish'...
8) waffles is 'elfish'...

End of the test

**************************************************
```

```
**********************************************************
QUESTION 3 PART 2 TEST:

1) elfish is 'elfish'...
2) pencil is not 'elfish'...
3) whiteleaf is 'elfish'...
4) freq is not 'elfish'...
5) tasteful is 'elfish'...
6) rampage is not 'elfish'...
7) unfriendly is 'elfish'...
8) waffles is 'elfish'...

End of the test

**********************************************************
QUESTION 3 PART 3 TEST:

1)The array is:
64 25 12 22 11 Sorted...
Result:
11 12 22 25 64

2)The array is:
12 11 13 5 6 Sorted...
Result:
5 6 11 12 13

3)The array is:
-5 -10 0 -3 8 5 -1 10 Sorted...
Result:
-10 -5 -3 -1 0 5 8 10

4)The array is:
40 60 1 200 9 83 17 -5 -10 0 -3 8 5 -1 10 Sorted...
Result:
-10 -5 -3 -1 0 1 5 8 9 10 17 40 60 83 200

End of the test

**********************************************************
```

```
**************************************************
QUESTION 3 PART 4 AND 5 TEST:


Postfix expression: 10 50 5 2 * - 10 / + 2 + 21 7 / -
Result: 13

Postfix expression: 8 2 * 3 * 3 - 8 4 / 1 1 + / +
Result: 46

Postfix expression: 2 3 1 * + 9 -
Result: -4

Postfix expression: 12 2 + -
null
This is a not postfix expression!!!

Prefix expression: - + * 2 3 * 5 4 9
Result: 17

Prefix expression: - + + 10 / - 50 * 5 2 10 2 / 21 7
Result: 13

Prefix expression: + - * 2 2 / 16 8 5
Result: 7

Prefix expression: - 10 2 +
null
This is a not prefix expression!!!

End of the test
**************************************************
QUESTION 3 PART 6 TEST:

1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16

Spiral of this array: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16
17 18 19 20

Spiral of this array: 1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10

1  2  3  4  5  6
7  8  9  10 11 12
13 14 15 16 17 18

Spiral of this array: 1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10


End of the test
**************************************************
```