CSDS 233 Assignment #1

Due Sep. 18th, 2020 before 11:59 p.m., 100 points

Written Exercises (50 pts)

1. Read the following code snippets. For each snippet, you need to: **[1]** provide its worst-case running time in the big-O format. Remember to keep it as tight as possible but do not chase constants (they do not matter as mentioned in the lecture). (5 pts each) **[2]** Explain your reason for your answer. (6 pts each)

   (a)

```
boolean foo(int[] a, int val) {
    for (int i = 1; i < a.length; i ++){
        if (a[i] == -1){
            a[i] = val;
            return true;
        }
    }
    return false;
}
```

   (b)

```
import java.lang.Math;
import java.util.Arrays;

int foo(int n){
    boolean isPrime[] = boolean[n+1];
    Arrays.fill(isPrime, true);

    for (int i = 1; i <= n; i ++)
        for (int j = 2; j <= Math.sqrt(i); j ++)
            if (i % j == 0)
                isPrime[i] = false;

    int sum = 0;
    for (int k = 1; k <= n; k ++)
```

```
        sum += isPrime[k] ? k : 0;
    return sum;
}
```

2. Determine whether the statement is true and prove or disprove it. (3 pts each)

   (a) $6n^2 + 13n - 6 = O(n^4)$

   (b) $n^2 - 2n + 4 = \Omega(n)$

3. Please simplify (as much as possible) the following big-O expressions: (3 pts each)

   (a) $O(2n^3 + 6n + 25n)$

   (b) $O(\log_2 n^2 + (\log_2 n)^2 + \ln n + 21)$

   (c) $O((n+2)^4 + (n+3)^2)$

4. Read the snippet below for computing the n-th Fibonacci number. **[1]** Will this program work? Explain your reason. If so, please draw a brief graph to show how it works considering its input `n = 4`. If not, please correct the code. (5 pts) [2] Then rewrite the code using iteration instead of recursion. (8 pts)

```
int fib(int n){
    if (n <= 1)
        return n;
    return fib(n + 1) + fib(n + 2);
```

Programming Exercise (50 pts)

In this part, you need to create two classes: `Course` and `CourseList`. `Course` stores information of a course, and `CourseList` stores a list of courses.

The `Course` class should contain those fields:

- `courseID`: a string, the ID of the course;
- `courseName`: a string, the complete name of the course;
- `capacity`: an integer, the initial capcacity limit of this course.

The `CourseList` class should contain a field and several functions:

- `listOfCourses`: an array of Course objects. You should only use `Arrays` (the built-in one) instead of `ArrayList`. For simplicity, you can use a large constant (such as `10`) as the maximum size of this list;
- `int size()`: returns the current size of the list, which is the number of the courses in the list;
- `void addCourse(int i, Course course)`: adds a new Course object before the `i-th` element of the list (the index of the first element is 0). If `i` is greater than the list size, adds it to the end;
- `boolean removeCourse(int i)`: deletes the `i-th` element of the list and returns true. If the list has less than `i` elements, returns false;
- `boolean changeCapacity(String courseID, int capacity)`: changes the capacity of the course with this courseID if this course is in the list and then returns true (which means it is a successful operation). Otherwise, returns false and do nothing instead;
- `Course getCourseWithIndex(int i)`: returns the `i-th` element of the list. If the list has less than `i` elements, returns null;
- `int SearchCourseID(String courseID)`: return the index of the course with this courseID in the list. If the course is not in the list, return `-1` instead;
- `int SearchCourseName(String courseName)`: return the index of the course with this courseName in the list. If the course is not in the list, return `-1` instead;

Additional requirements:

- Print out your list with neat format before and after each operation. You do not need to follow the format below. These are only examples.

  An acceptable output for an addCourse() function can be like:

```
Operation: Add a course to index 1.
Course: courseID: EECS233, courseName: Data Structures, capacity: 50

List before the operation:
0. courseID: CSDS132, courseName: Introduction to Java, capacity: 30
1. courseID: CSDS302, courseName: Discrete Mathematics, capacity: 80
2. courseID: CSDS310, courseName: Algorithms, capacity: 75

List after the operation:
0. courseID: CSDS132, courseName: Introduction to Java, capacity: 30
1. courseID: CSDS233, courseName: Data Structures, capacity: 50
2. courseID: CSDS302, courseName: Discrete Mathematics, capacity: 80
3. courseID: CSDS310, courseName: Algorithms, capacity: 75
```

Another example for a failed changeCapacity() function can be:

```
Operation: Change the capacity of the course to 15.
Course: courseID: EECS233, courseName: Data Structures, capacity: 50

List before the operation:
0. courseID: CSDS132, courseName: Introduction to Java, capacity: 30
1. courseID: CSDS302, courseName: Discrete Mathematics, capacity: 80

List after the operation:
0. courseID: CSDS132, courseName: Introduction to Java, capacity: 30
1. courseID: CSDS302, courseName: Discrete Mathematics, capacity: 80
```

Submission

The submissions will be evaluated on completeness, correctness, and clarity. Please provide sufficient comments in your source code to help the TAs read it. Please generate a single zip file containing all your *.java files needed for this assignment and optionally a README.txt file with an explanation about added classes and extra changes you may have done. Name your file P1_YourCaseID_YourLastName.zip. Submit your zip file electronically to Canvas.

Grading:

- Course implementation: 10 pts
- CourseList implementation: 15 pts
- Customized Demo with all functions included: 10 pts

- Proper encapsulation/information hiding: 5 pts
- Design and style: 10 pts
  - Style: 3 pts.
  - Comments: 4 pts.
  - Design: 3 pts.