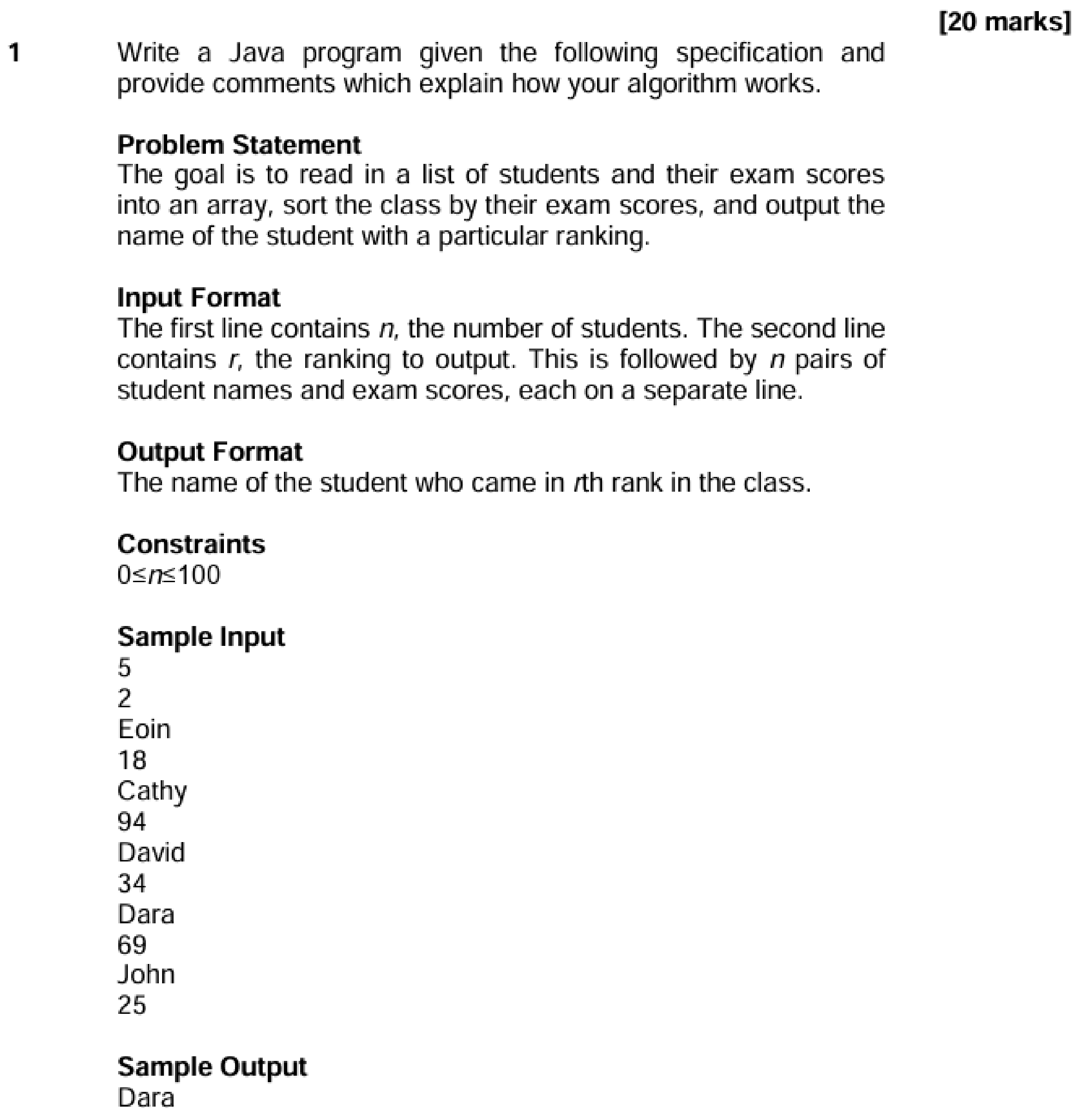
Question 1



[20 marks]

1 Write a Java program given the following specification and provide comments which explain how your algorithm works.

Problem Statement
The goal is to read in a list of students and their exam scores into an array, sort the class by their exam scores, and output the name of the student with a particular ranking.

Input Format
The first line contains *n*, the number of students. The second line contains *r*, the ranking to output. This is followed by *n* pairs of student names and exam scores, each on a separate line.

Output Format
The name of the student who came in *r*th rank in the class.

Constraints
0≤*n*≤100

Sample Input

```
5
2
Eoin
18
Cathy
94
David
34
Dara
69
John
25
```

Sample Output

```
Dara
```

```java
import java.util.Scanner;
import java.util.Queue;
import java.util.PriorityQueue;

public class Q1 {
   public static void main (String args[]) {
      Scanner sc = new Scanner(System.in);
      Queue<Student> pq = new PriorityQueue<Student>();
      // number of students
      int SIZE = Integer.parseInt(sc.nextLine());
      // output rank
      int RANK = Integer.parseInt(sc.nextLine());

      // Create a new Student class, input its name and score
      for(int i = 0; i < SIZE; i++) {
         String inputName = sc.nextLine();
         int inputScore = Integer.parseInt(sc.nextLine());
         pq.add(new Student(inputName, inputScore));
      }
      sc.close();

      for(int i = 1; i < RANK; i++) {
          pq.poll();
      }
      System.out.println(pq.poll().name);
   }
}

// Create a new Class - Student
// It has two properties - name, score
class Student implements Comparable<Student>{
   String name;
   int score;

   public Student(String name, int score) {
      this.name = name;
      this.score = score;
   }

   //Higher score has priority, when use PriorityQueue
   @Override
   public int compareTo(Student other) {
      return other.score - this.score;
   }
}
```

CS210-2018-Autumn

Question 2

[20 marks]

2 Write a Java program given the following specification and provide comments which explain how your algorithm works.

Problem Statement
If you flip a single coin, you have a 50% chance of getting a single tail. If you flip two coins, you now have a 75% of seeing at least 1 tail. What is the chance you will see at least *T* tails after *N* coin tosses? Use a Monte Carlo simulation and round to the nearest percent.

Input Format
The first line is an integer *N*, the number of coin tosses. The second line in is an integer *T*, the target number of tails.

Output Format
An integer from 0 to 100 representing the percentage probability that at least *T* tails will be observed given *N* tosses of a fair coin.

Constraints
$0 \le N \le 1000$
$0 \le T \le 1000$

Sample Input
4
1

Sample Output
94

(if you flip 4 coins, the probability of seeing at least 1 tail is 93.75%)

```java
import java.util.Scanner;

public class Q2 {
   public static void main (String args[]) {
      Scanner sc = new Scanner (System.in);

      // input number of coins tosses
      int TOSSES = Integer.parseInt(sc.nextLine());
      // input target number of tails
      int TAILS = Integer.parseInt(sc.nextLine());
      sc.close();
      //Monte Carlo
      int N = 1000000;
      int count = 0;

      for(int i = 0; i < N; i++) {
         int tailToss = 0;
         for (int j = 0; j < TOSSES; j ++) {
            /* create random number: 0 or 1
              50% will be 0 -> coin tails
              50% will be 1 -> coin head */
            int status = (int) (2 * Math.random());
            if(status == 0) tailToss ++;
         }
         if (tailToss >= TAILS) count++;
      }

      double p = (double) 100 * count / (double) N;
      // Round up to the nearest Integer 93.75 -> 94
      System.out.println(Math.round(p));
   }
}
```

Question 3

[20 marks]

3 Write a Java program given the following specification and provide comments which explain how your algorithm works.

Problem Statement
Manipulate a stack according to the given push and pop commands and then output the number that is at the top of the stack. If a pop command is issued for an empty stack then nothing should happen.

Input Format
The first line is a number *N*, which indicates the number of commands to follow. This is followed by *N* lines, each of which consists of the word PUSH or POP. The word PUSH will be followed by an integer *n*.

Output Format
Output the integer that is at the top of the stack following the given commands. If the stack is empty then output "empty".

Constraints
$1 \leq N \leq 10$
$-10000 \leq n \leq 10000$

Sample Input
```
5
PUSH 4
PUSH 8
POP
POP
PUSH 2
```

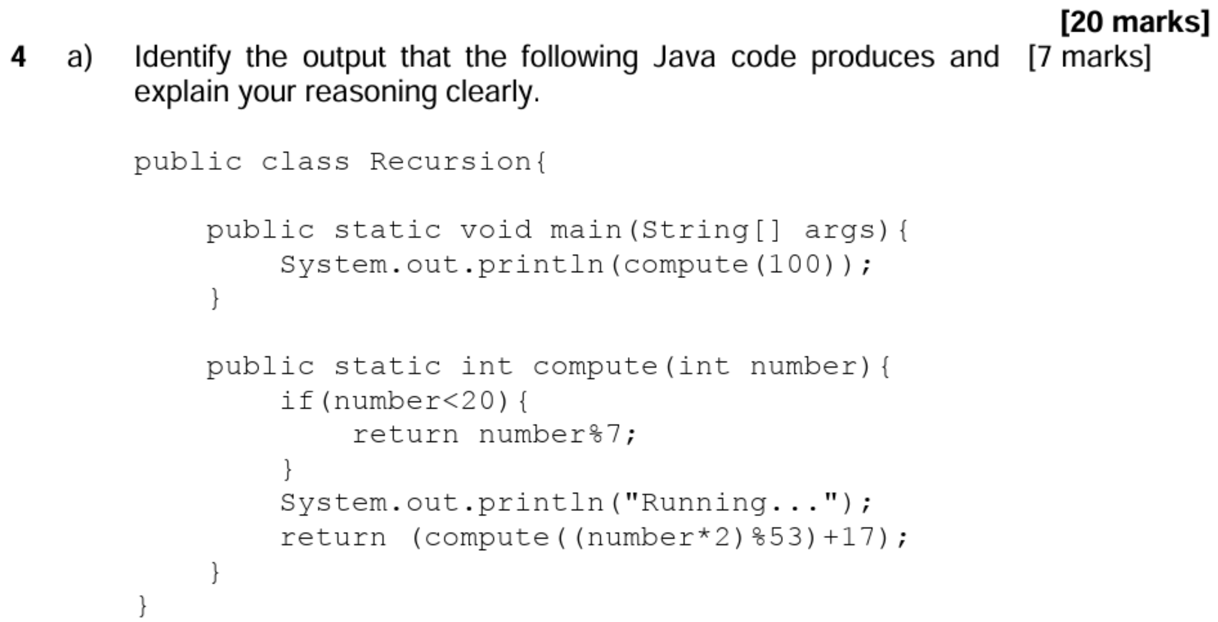
Sample Output
```
2
```

```java
import java.util.Scanner;
import java.util.Stack;

public class Q3 {
    public static void main (String arg[]) {
        Stack<Integer> s = new Stack<Integer>();
        Scanner sc = new Scanner(System.in);
        // Number of Commands
        int N = Integer.parseInt(sc.nextLine());
        for(int i = 0; i < N ; i++) {
            String inputLine = sc.nextLine();

            //PUSH Commands
            if(inputLine.split(" ")[0].toUpperCase().equals("PUSH")) {
                int inputNum = Integer.parseInt(inputLine.split(" ")[1]);
                s.add(inputNum);
            }

            //POP Commands
            if(inputLine.toUpperCase().equals("POP")) {
                //When stack is empty, nothing happens.
                if(!s.isEmpty()) s.pop();
            }
        }
        sc.close();

        // If stack is empty, then output "empty"
        if(s.isEmpty()) {
            System.out.println("empty");
        }// output the Integer at the top of the stack
        else {
            System.out.println(s.peek());
        }
    }
}
```

CS210-2018-Autumn

Question 4

Question a



[20 marks]

4 a) Identify the output that the following Java code produces and explain your reasoning clearly. [7 marks]

```
public class Recursion{

    public static void main(String[] args){
        System.out.println(compute(100));
    }

    public static int compute(int number){
        if(number<20){
            return number%7;
        }
        System.out.println("Running...");
        return (compute((number*2)%53)+17);
    }
}
```

The program runs main function first, it will call `compute(100)`

1) compute(100).

100>20=> skip if statement

Print "Running...",then change line

return (compute((100 * 2) % 53) + 17) = compute(41) + 17

2) compute(41).

41>20=> skip if statement

Print "Running..." ,then change line

return (compute((41 * 2) % 53) + 17) = compute(29) + 17

3) compute(29).

29>20=> skip if statement

Print "Running..." ,then change line

return (compute((29 * 2) % 53) + 17) = compute(5) + 17

4) compute(5).

5<20=> run if statement

return 5 % 7 => compute(5) = 3

CS210-2018-Autumn

5) Calling compute(29)
   compute(29) = compute(5) + 17 = 5 + 17 = 22

6) Calling compute(41)
   compute(41) = compute(5) + 17 = 22 + 17 = 39

7) Calling compute(100)
   compute(100) = compute(41) + 17 = 39 + 17 = 56

Therefore, the Java Program outputs

`

```
  Running
  Running
  Running
  56
```

`

when it runs.

CS210-2018-Autumn

Question b

b) Identify the output that the following Java code produces and explain your reasoning clearly. [7 marks]

```
public class BitManipulation{

    public static void main(String[] args){
        System.out.println((((4|17)|2))>>1);
    }
}
```

The program will print out the equation

((4|17)|2)>>1)

Step 1: 4 & 17

$(4)_{10}$ =$(00000100)_2$

$(17)_{10}$ =$(00010001)_2$ |

`$(00010101)_2$ = $(21)_{10}$

Step 2: 21 | 2

$(21)_{10}$ =$(00010101)_2$

$(2)_{10}$ =$(00000010)_2$ |

`$(00010111)_2$ = $(23)_{10}$

Step 5: 23 >> 1

$(00010111)_2$ >> 1 = $(00001011)_2$ = $(11)_{10}$

Therefore, the Java Program outputs 11 when it runs.

*Question c

c) Show how the following numbers would be sorted by mergesort. [6 marks]
State the **Big O complexity** of mergesort and explain why it is more efficient than bubble sort.

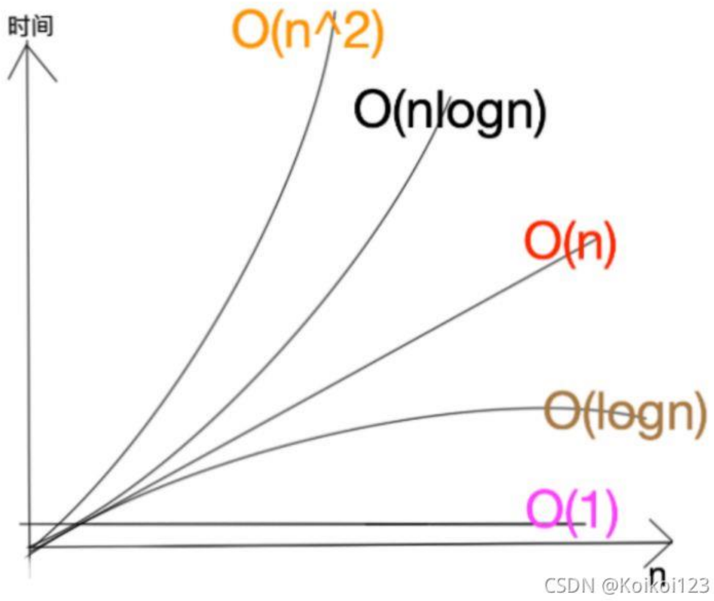
33 63 90 68 21 96 38 27

[33, 63, 90, 68, 21, 96, 38, 28]

- Divide the list: [33], [63], [90], [68], [21], [96], [38], [27]
- Merge pairs and sort: [33, 63], [68, 90], [21, 96], [27, 38]
- Merge sublists and sort: [33, 63, 68, 90], [21, 27, 38, 96]
- Merge the two sorted sublists: [21, 27, 33, 38, 63, 68, 90, 96]

Finally, sorted list: [21, 27, 33, 38, 63, 68, 90, 96]

The Big O Complexity of mergesort is O(n * log(n))
The Big O Complexity of bubblesort is $O(n^2)$
As we can see in the graph, if n is the same value, $O(n^2) > O(n * \log(n))$.
Therefore, mergesort use less time, it is more efficient than bubblesort.



CS210-2018-Autumn