

King Fahd University of Petroleum & Minerals  
Information and Computer Science Department  
ICS 201 – Introduction to Computing II  
First Semester 2011-2012 (111)

Name:

ID#:

**Final Exam**

Time allowed: 120 minutes

Sunday, January 8<sup>th</sup>, 2012

*Please circle your section number below*

Section	01	02	03	05
Instructor	Zhioua	Irfan	Sukairi	Sukari
Day and Time	SMW 08-08:50	SMW 09-09:50	SMW 10-10:50	SMW 13:10-14:00

**Notes:**

1. The exam has **FIVE** questions and consists of **NINE** pages.
2. You are expected to answer all questions.
3. This is a closed book exam.
4. Please free to use the back of the page. However, please make sure you indicate this in order for me to not miss it for grading.

Question #	Maximum Mark	Obtained Mark
1	10	
2	30	
3	10	
4	20	
5	30	
<b>Total</b>	<b>100</b>	

**Question 1: [10 points]**

a) [2 points] Why can't an abstract method be declared private?

b) [4 points] To use a constructor to create instance of an abstract class is illegal. So why bother to have any constructors in abstract class? It seems useless. Explain.

c) [4 points] Which of the following is true [select all that apply]:

- A final method cannot be overridden.
- A final class cannot be instantiated.
- A final instance variable should always be static.
- A final class cannot be inherited.



```

void printStar(int i) {
    for (int x = 1; x<=i; x++)
        System.out.print("*"); }

//recursive function printPattern
//You need to decide the method header/parameter(s)

```

c) [10 points] What will be the output of the following program:

```

public class FinalExam {
    int curCount;

    public FinalExam() {
        curCount = 1;
        Thread cu = new CounterUp();
        Thread cd = new CounterDown();
        cu.start();
        cd.start();
    }
    private class CounterUp extends Thread{
        public void run(){
            for (int i = 1; i <=4; i++){
                try{
                    sleep(40);
                }catch(Exception e){}
                curCount = curCount + 1 ;
                System.out.println(curCount);
            }
        }
    }
    private class CounterDown extends Thread{
        public void run(){
            for (int i = 1; i <=4; i++){

```

```
        try{
            sleep(100);
        }catch(Exception e){}
        curCount = curCount - 1 ;
        System.out.println(curCount);
    }
}
public static void main (String args[]){
    FinalExam f = new FinalExam();
}
}
```

OUTPUT

### Question 3 [10 points]

- a) Assuming that the array parameter is always sorted in ascending order (from smaller to larger); can the linear search algorithm be improved? If no explain why, if yes give the code for the improved linear search method (do not provide binary search algorithm).

Assume that you have the following sorted array:

Index	0	1	2	3	4	5	6	7	8	9	10	11
Value	12	17	23	37	39	44	57	67	73	77	85	96

The binary search algorithm uses three indices: **Left** (or l), **Right** (or r) and **Mid** (or m). These indices are updated at each iteration (or recursive call). Show the values of Left, Right, and Mid at each step if the target value is **37** as shown below:

Left	Right	Mid

#### Question 4 [20 points]

- a) Does the initial position of the elements of an array have an impact on the performance of Selection Sort algorithm? Justify your answer.

**Clarification:** By initial position of the elements we mean how the elements are initially placed in the array.

- b) What is the main advantage of QuickSort with respect to MergeSort?

**Consider the following array:**

57	-5	14	88	8	-20	34	76	11	60	19	8	22
----	----	----	----	---	-----	----	----	----	----	----	---	----

c) Apply MergeSort and show the steps of the algorithm

d) Show the different steps of applying method partition on the above array.

**Question 5 [30 points]**

An online dictionary is needed. The user will enter a word in a language and the program will produce a list of "translations" in another language. Note that some words, like "Java", have several translations. It's necessary to store multiple translations for a given word.

Assume the following data structure as a design:

```
public class MyDictionary {
    private ArrayList<DicEntry> dictionary = new ArrayList<DicEntry>();

    private class DicEntry {
        private String word;
        private ArrayList<String> transList;

        public DicEntry(String w, ArrayList<String> t) {
            word = w;
            transList = t;
        }

        public String getWord() {
            return word;
        }

        public ArrayList<String> getTransList() {
            return transList;
        }
    }

    public ArrayList<DicEntry> getDictionary() {
        return dictionary;
    }

    public void AddWord(String word, String definition) { //code for (a)

}
}
```

```
public void translate(String word) { //code for (b)

}

public String maxDef() { //code for (c)

}

}
```

- a) Write a method, `public void AddWord(String word, String definition)`, which associates the definition with the word. It will add *definition* (translation) to the collection of existing definitions for this word (or create the first entry if there isn't one).
- b) Write a method, `public void translate(String word)`, which displays a list of translations for a word if it is found in the dictionary, or "word not found" otherwise. Linear search of the dictionary is accepted.
- c) Write a method, `public String maxDef()`, which returns the word with maximum number of definitions in the dictionary. If more than one word have the same maximum, return any.