University of Illinois at Urbana-Champaign
Department of Computer Science

Third Examination

CS 125 Introduction to Computer Science
90 minutes permitted

Name: _____

NetID: _ _ _ _ _ _ _ _ _
(this is the username part of your @illinois.edu email address – please write legibly)

Lab Section (Day/Time & Code): _____ _____

• The exam proctors will not answer any technical questions. If you believe a question is ambiguous, write your assumptions on your sheet and answer accordingly.

• This is a closed book and closed notes exam. No electronic aids are allowed.

• This exam tests your understanding of recursion: **Unless specifically instructed you may not use loops (for/while/do-while) in this exam**.

• You are not allowed to use the break, continue, or switch statements on this exam.

• Unless we say otherwise in the specific problem, you can assume all values entered by the user will be acceptable input for that program.

• When you write code, you may use a shorthand for System.out and TextIO input and output methods provided it is obvious to the graders which method you are using. For example it is acceptable to use Sopln in place of System.out.println and to use Sopt in place of System.out.print Likewise, you can use T.rlnI(), T.rlnC(), and T.rlnD() in place of TextIO.readlnInt(), TextIO.readlnChar(), and TextIO.readlnDouble().

• For full marks correct syntax is required: Ensure all statements include a semicolon and the correct use of upper/lower case, single quotes and double quotes.

Problem	Points	Score	Grader
1	10		
2	15		
3	15		
4	15		
5	15		
6	15		
7	15		
Total	100		

1. Recursive Concepts – 10 points (2 points each)

Consider the following code that is supposed to recursively count how many ways a particular spaghetti brand can break into two.

```
public static int split(int len) {

     // Split even spaghetti into two pieces:
     if( len % 2 == 0 )
        return split(len/2) + split(len/2);

    // else split odd spaghetti by one segment:
     return 1 + split(len -1);
}
```

a. What is the most significant oversight / error in the above code?

b. When would you expect to see a problem - When you compile and / or execute in a Java Virtual Machine? (Circle the correct answers)

i) Compile Problem? Yes / No

ii) Execution Problem? Yes / No

c. Is the following tail- or forward- recursive? Briefly explain your answer.

```
int count(int x) {
    if(x<=0) return 0;
    return 1 + count(x - 1);
}
```

d. Circle and briefly explain **one** of the following CS125 terms:

Unit test

Single step debugging

Checkout

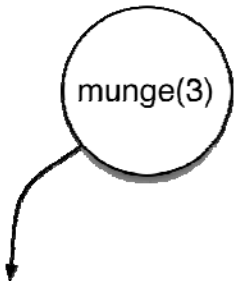
e. In MP7, two DNA sequences are represented as character arrays and recursively compared using which conceptual idea? (Circle the correct answer)

Shortest Common Dictionary	Longest Common Subsequence
Longest Alpha Chain	Greatest Ribonucleotide Factorial

2. Tracing code – 15 points
Consider the following method:

```
public class SecretFunction {
   public static int munge(int x) {
      if (x == 0 || x == 1) {
         return 1;
      }
      if (x < 0) {
         return 2 * munge(-x/2);
      }
      return x + munge(x/2) + munge(2-x);
   }
}
```

a. Draw the activation diagram for the execution of `munge(3)` and answer the two questions below. The activation diagram should include nodes (with the method parameter) for each execution of the `munge` method, and arcs between nodes for calls and returns. Return arcs should include the returned value.



b. Result of **`munge(3)`** ? _____

c. How many times is **`munge`** activated (called), including `munge(3)` ?_____

3. Linked Lists – 15 points

Complete the following `LinkedList` class by writing two recursive instance methods:

a. **`isListSorted`:** returns **true** if for *all links in the list* the value of the next link is equal or greater than the current link; returns **false** otherwise.

b. **`countAboveThreshold`:** count the number of items in the linked list whose "value" is larger than the provided threshold.

Do not use any loops in this class!

```
public class LinkedList {
   private double value;
   private LinkedList next;

   public boolean isListSorted() {









   }
   public int countAboveThreshold(double threshold) {









   }
  … Constructor code not shown.
} // End of LinkedList class
```

4. Maze Exploration – 15 points

Read the given code and comments below then create the recursive class method 'count' to return the total number of routes from position (x,y) to position (tx,ty). Determine the method type, parameters and return type from the given code. A route consists of zero or more 'Compass' moves: Each move can increment or decrement the x or y value by one but not both at the same time. Valid routes do not visit a blocked square (`walls[x][y]` is true), or visit locations outside of (0,0) …(size-1,size-1), or visit the same (x,y) square twice as part of the same route. Assume the start and end locations are not blocked by a wall.

```
public class Maze {
  public static void main(String[] args){
    int size = …; // Assume 1 < size <100
    boolean[][] walls = generateMaze(size);
    // if wall[x][y] is true then (x,y) is a wall and not part of a route.
    boolean[][] blocked = new boolean[size][size];
    // The array 'blocked' is used to prevent infinite recursion-
    // i.e. Do not recursively explore part of the current route.
    int x =… , y=… , tx=… , tz= … ;
    // Assume variable values are between 0 … size-1
    int result = count(x,y, tx,ty, walls,blocked);
    System.out.println("Number of routes:"+result);
  }
  public static boolean[][] generateMaze(int size) {
    boolean[][] result = new boolean[size][size];
    // The four edges are always continuous walls:
    for(int i=0;i<size;i++) {
      result[i][0] = result[i][size-1] = true;
      result[0][i] = result[size-1][i] = true;
    }
    Rest of generateMaze method not shown …
  }
```

5. Binary Search – 15 points
You need to search a sorted array of `Person` objects.

```
public class Person{
     private int age;
     private String name;
     public int getAge() {return age;}
     public String toString() {return name+"("+age+")"; }
     … Constructor code not shown.
}
```

a. Complete the following **recursive binary search** method to quickly find and return a Person of a particular **age**. Use a 'divide and conquer' approach by exploiting the fact that the array is sorted by increasing age. All people have unique ages. Search the array only between lo^{th} and hi^{th} indices. Return '`null`' if no person matches the search age. All entries in the array are valid and non-null.

Notice the search method is in a different class than Person. Do not use loops.

```
class Util {
   public static Person search(Person[] people, int age, int lo,
                               int hi) {
```

b. Create a non-recursive public class method `exists` that takes two parameters - an array of `Person` objects and an `int`, the search age. This method delegates most of the computation to the recursive method `search` above and returns true if the array contains a person of the given age, false otherwise.

Write your method here:

6. Recursive Searching and Sorting Concepts – 15 points

a. Complete the following recursive method to return the index of the smallest value of all entries `data[lo]`, `data[lo+1]` up to and including `data[hi]`.
Do not use any loops. The data is not sorted. Assume 0 <= lo <= hi < data.length and the array values are distinct.

```
public static int findMin(double[] data, int lo, int hi) {




}
```

b. For the following code how many times is `findMin` activated; *i.e.* how many times is it called, including the recursive cases and the call below?

```
double[] d1000 = { 5., 3. , 11., 1., 10, 11, … 994 more entries };
int pos = findMin(d1000, 0, 5);
```

Your Answer: _____

c. Which one of the following best describes the Selection Sort algorithm?

A. Using a pivot value partition the array into two sub arrays (all values lower and all values higher than the pivot value)
B. Divide the array into two: Recursively sort the two pieces independently then merge together the two sorted sub arrays into one.
C. Find the next smallest value from the unsorted values and move it to the end of the sorted values.
D. Take the next value from the unsorted values and insert it into the correct position, moving previously sorted values to make room.

Your Answer: _____

d. Consider the following array of 10 values for sorting using Selection sort (low to high).

6	2	10	16	4	52	54	7	60	58

Calculate the values in the array after the 4th swap has completed. Write your answer below:

e. Once all of the values have been sorted and all swaps have completed,

How many times has the value '6' moved to a new position? _____

How many times has the value '60' moved to a new position? _____

7. Sorting Implementation – 15 points

The following code has many errors and omissions. Fix the three methods below to correctly implement a recursive selection sort so the code matches the behavior described in the comments. You may assume the `findMin` method is correctly implemented.

```
/** A wrapper method to sort the entire array.*/
public static void sort (double[] data) {




}
/** Sorts all data values between lo and hi (inclusive) using a recursive
selection sort.*/
public static void sort(double[] data, int lo, int hi) {


   // findMin(double[], int lo, int hi) is implemented on the previous page
   int minPos = findMin(data,









}
```

```
/** Swaps values data[i] and data[j] */
public static void swap(double[] data, int i, int j) {


   data[i] = data[j];


   data[j] = data[i];


}
```

END OF CS125 EXAM
CHECK YOUR WORK
ENSURE YOUR NETID IS ON EVERY PAGE