Name _____

CISC 3115 – Modern Programming Techniques
Fall 2024
Exam 2
Solutions

Answer 5 out of 6 of the following questions.

1. The method

```
void lookup(String u, String p)
```

does a search of some array of pairs to see if the pair (u, p) appears (you don't have to worry about the mechanics of this method). If it doesn't appear, it throws a `PairNotFoundException` with the message "Not found"; otherwise, it simply returns to the caller.

What does the following code do? For full credit, describe it in terms of its application.

```
int count = 0;
System.out.print("u? ");
String u = keyboard.next();
System.out.print("p? ");
String p = keyboard.next();
while (true) {
   try {
         lookup(u, p);
         break;
   } catch (PairNotFoundException e) {
         System.out.println(e.getMessage());
         count++;
         if (count == 3) {
               System.out.println("Bye");
               System.exit(1);
         }
         else {
               System.out.print("u ? ");
               u = keyboard.next();
               System.out.print("p? ");
               p = keyboard.next();
         }
   }
```

The user is prompted for a pair of strings which is looked up in a database. If unsuccessful, an informative message is printed. After three unsuccessful searches the programs terminates, otherwise the code continues. This would be useful in a user/password, i.e., login protocol (such as presented in Lab 4).

A line-by-line translation of the Java to English (e.g. *prompt for two string, enter a loop, call the method, if an exception occurs, print out a message, add 1 to count. If count is 3 print "Bye" and leave the program, etc*) is insufficient.

2.

a. Write a method

```
int getExamGrade(Scanner sc)
```

that returns the next (integer) data item from `sc`. If the next item is not an integer, or its value is less than 0 or greater than 100, throw a `DataException` with an appropriate message, i.e., "non-integer input", or "grade out of range" (depending on the error).

For example, if the next item in the file was:

- 78, the program should return 78
- if the next item in the file was "Hello", the method should throw a `DataException` with the message "non-integer input"
- if the next item in the file was 105, the method should throw a `DataException` with the message "grade out of range"

```
int getExamGrade(Scanner sc) throws DataException {
    if (!sc.hasNextInt())
        throw new DataException("non-integer input");

    int num = sc.nextInt();
    if (num < 0 || num > 100)
        throw new DataException("grade out of range");

    return grade;
}
```

I forgot to put the throws DataException on the method header … wasn't expecting you to add it.

b. Write a fragment of code that creates a `Scanner` object on the file `roster.text`, containing lines of student data consisting of a name followed by an integer header value, followed by that many grades. For each student, if all the grades are valid (each input and checked using `getGrade`), print the name and average; otherwise print the exception message (and go on to the next student). Assume each header value contains the correct number of items that follow.

For example, if the input contained:

```
Weiss   2   90   80
Arnow   3   75   A23 95
Sokol   4   80   90   105   70
```

The output would be:

```
Weiss   85
Arnow   non-integer input
Sokol   grade out of range
```

```
Scanner sc = new Scanner(new File("roster.text"));

while (sc.hasNext())
      try {
            String name = sc.next();

            int n = sc.nextInt();
            int total = 0;
            for (int i = 1; i <= n; i++) {
                  total += getGrade(sc);

            double average = (double)total / n;
            System.out.println(name + " " + average);
      catch(DataException e) {
            System.out.println(e.getMessage());
      }
}
```

3.

a. Write the method,

```
boolean binsearch(int [] arr, int lo, int hi, int val)
```

that performs a recursive binary search for `val` in `arr` (between `lo` and `hi`).

```
boolean binsearch(int [] arr, int lo, int hi) {
     if (lo > hi) return false;

     int mid = (hi + lo)/2;

     if (val == arr [mid])
        return true;
     else if (val < arr[mid])
        return binsearch(arr, lo, mid-1);
     else
        return binsearch(arr, mid+1, hi);
}
```

b. Write the wrapper method

```
boolean search(int [] arr)
```

that a user would call to invoke this search (assume a fully populated array, i.e., the entire array is to be searched.).

```
boolean binsearch(int [] arr) {
    return binsearch(arr, 0, arr.length-1);
}
```

4. Given the following array:

```
int a[] = {1, 2, 3, 2, 1};
```

and the methods:

```
int m1(int [] arr, int lo, int hi, int val) {
      if (lo > hi) return -1;
      if (arr[lo] == val) return lo;
      return m1(arr, lo+1, hi);
}

int m2(int [] arr, int lo, int hi, int val) {
      if (lo > hi) return -1;
      if (arr[hi] == val) return hi;
      return m2(arr, lo, hi-1);
}
```

a. What is returned by the call `m1(arr, 0, arr.length-1, 2);`?

1

b. What is returned by the call `m2(arr, 0, arr.length-1, 1);`?

4

c. What do the methods do? How are they different?

Both search the array for the specified value. m1 searches from the beginning (i.e., for the first occurrence); m2 from the end (last occurrence).

d. What is wrong with the following version?

```
void m3(int [] arr, int lo, int hi) {
      If (lo > hi) return -1;
      If (arr[lo] == val) return lo;
      return m1(arr, lo, hi-1);
}
```

Compares the value against the element at the front but if doesn't find it, recurses from the end, i.e., 'chops of the last element rather than the first.

5.

a. Here is a recursive method to test if (the non-negative integers) `x` and `y` are equal

```
boolean equals(int x, int y) {
    if (x == 0 && y == 0) return true;
    if (x == 0 || y == 0) return false;
    return equals(x-1, y-1);
}
```

Write a similar recursive method:

```
boolean notEquals (int x, int y)
```

that returns whether `x` is <u>not</u> equal to `y`. You may not call `equals` in your definition of `notEquals`.

```
boolean equals(int x, int y) {
    if (x == 0 && y == 0) return false;
    if (x == 0 || y == 0) return true;
    return equals(x-1, y-1);
}
```

b. 0 is an even number. A positive integer `x` is even if `x-1` is not even (i.e. odd). Write the recursive method `boolean isEven(int x)`

```
boolean isEven(int x) {
    if (x == 0) return true;
    return !isEven(x-1);
}
```

6.

a. How many escape clauses does the recursive method for factorial have and what are they?

1 escape clauses: x = 0

b. How many escape clauses does the recursive method for the Fibonacci sequence have and what are they?

2 escape clauses: x = 0 and x = 1 (x <= 1)

c. Explain why the number of escape clauses are different or the same.

Factorial makes one recursive call to x-1, eventually all numbers work their way down to 0

Fibonacci has a recursive call to n-1 and another to n-2. If the only escape clause was 0, when the number works its way down to 1 (not an escape clause), one of the recursive call made would be to n-2, or -1, thus skipping (i.e., 'missing') the escape clause, and there'd would be a runaway recursion in the negative range.