German University in Cairo
Media Engineering and Technology Faculty
Prof. Dr. Slim Abdennadher

April 23, 2014

CSEN202: Introduction to Computer Programming
Spring Semester 2014

Midterm Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Duration of the exam: 2 hours (120 minutes).
2) No books or other aids are permitted for this test.
3) This exam booklet contains 13 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
4) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the four extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
5) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below ;-)

Exercise	1	2	3	4	5	6	$\sum$
Possible Marks	14	6	10	12	14	10	66
Final Marks							

Exercise 1 (1+2+2+2+2+2+3=14 Marks)

a) How many bits are there in a `byte`?

Solution:

8 bits

b) How many different unique values can you represent with a `byte`? Give the smallest and largest value and justify your answer.

Solution:

2^8 different numbers since Java uses the 2's complement representation. The smallest number is -128 and the largest number is 127.

c) What's the output of the following program segment? **Hint**: Reformatting the code will help you understand the nesting levels.

```
boolean warm, windy;
String month;
warm = false;
windy = true;
month = "November";
if (warm)
if (windy)
if (month == "April")
System.out.println( "Maui" );
else System.out.println( "Perth" );
else System.out.println( "San Diego" );
else if (windy)
if (month == "April" )
System.out.println( "Santa Cruz" );
else System.out.println( "Peng Hu" );
System.out.println( "I need a break" );
```

Solution:

The output of the program is

```
Peng Hu
I need a break
```

The program with identation:

```
boolean warm, windy;
String month;
warm = false;
windy = true;
month = "November";
if (warm)
    if (windy)
       if (month == "April")
          System.out.println( "Maui" );
       else System.out.println( "Perth" );
    else System.out.println( "San Diego" );
else if (windy)
        if (month == "April" )
           System.out.println( "Santa Cruz" );
        else System.out.println( "Peng Hu" );
System.out.println( "I need a break" );
```

d) How many times will the method `doSomething()` be executed?

```
for( i = -5; i < 5; i +=2 )
      doSomething();
```

Solution:

Five times

e) What is the output of the following program segment?

```
incr = 200;
for( i = 5; i <= 10; i += incr )
{
     incr = -i;
     System.out.println( "incr is " + incr );
     System.out.println( "i is " + i );
}
```

Solution:

The program will print first

```
incr is -5
i is 5
```

and then prints an infinite number of

f) The following program segment is suppose to sort three numbers and print them in decreasing order.

```
int side1, side2, side3;
boolean vs12, vs13, vs21, vs23, vs31, vs32;
vs12 = (side1 > side2);
vs13 = (side1 > side3);
vs21 = (side2 > side1);
vs23 = (side2 > side3);
vs31 = (side3 > side1);
vs32 = (side3 > side2);
if (vs12 && vs13)
     if (vs23)
        System.out.println( side1 + side2 + side3 );
     else System.out.println( side1 + side3 + side2 );
else if (vs21 && vs23)
        if (vs13)
           System.out.println( side2 + side1 + side3 );
        else System.out.println( side2 + side3 + side1 );
     else if (vs12)
    System.out.println( side3 + side1 + side2 );
else System.out.println( side3 + side2 + side1 );
```

- Assume that `side1`, `side2`, and `side3` have the following values respectively: 33, 15, 27. What's the output of the above code segment?

  Solution:
  The program will print 75.

- Fix the code segment so that it produces the intended results.

  Solution:
  The program should print a string and not an integer:

```
System.out.println(side1 + " " + side2 + " " + side3);
```

Exercise 2 (6 Marks)

You would like to design a weekly workout plan for yourself. You decided for the following plan:

- Monday: Train your Upper Body
- Tuesday: Train your Lower Body
- Wednesday: Take off
- Thursday: Train your Upper Body
- Friday: Train your Lower Body
- Saturday: Take off
- Sunday: Take off

Write a Java method `activity` that given the day will return the activity that should be performed according to the above plan.

You method should use a switch statement and your code should be as short as possible, i.e. using the least number of statements. You are not allowed to use more than one `return` statement.

Solution:

```
public static String activity(String day) {
    String result = "";
    switch(day) {
    case "Monday":
    case "Thursday": result = "Train your Upper Body"; break;
    case "Tuesday":
    case "Friday": result = "Train your Lower Body"; break;
    case "Wednesday":
    case "Saturday":
    case "Sunday": result ="Take off"; break;
    default: result ="Wrong day";
    }
    return result;
}
```

Exercise 3 (10 Marks)

Write an interative Java method that takes two integers `x` and `y` and returns a new integer by alternating the digits of `x` and `y`. The digits should be alternated from the right to the left.

Your method should return the following for the calls:

```
merge(168,734);     -> 176384
merge(1,567);       -> 5617
merge(1786,34);     -> 178364
merge(0,0);         -> 0
merge(0,123);       -> 123
merge(123,0);       -> 123

public static int merge(int x, int y) {
```

Solution:

```
public static int merge(int x, int y)
{
    int r = 0;
    int i = 0;

    while(!(x==0) || !(y==0))
    {
       if(x==0)
     {
        r = r + (int)((y%10)*Math.pow(10,i));
        i++;
       }
       else
         if(y==0)
         {
               r = r + (int)((x%10)*Math.pow(10,i));
               i++;
          }
          else
          {
               r = r + (int)((y%10)*Math.pow(10,i)) + (int)((x%10)*Math.pow(10,i+1));
               i = i + 2;
          }
    x /= 10;
    y /= 10;
    }
return r;
}
```

Exercise 4 (12 Marks)

The longest palindromic substring problem is the problem of finding a maximum-length contiguous substring of a given string that is also a palindrome.

Write an Java method `longestPalindrome` that given a string `s`, it returns the longest palindromic substring.

For example, the longest palindromic substring of `"bananas"` is `"anana"`. The longest palindromic substring is not guaranteed to be unique; for example, in the string `"abracadabra"`, there is no palindromic substring with length greater than three, but there are two palindromic substrings with length three, namely, `"aca"` and `"ada"`. Thus, your method should return the first substring with the greatest length. In this case, `"aca"`.

```
public static String longestPalindrome1(String s) {
```

Solution:

```
public static String longestPalindrome1(String s) {

int maxPalinLength = 0;
String longestPalindrome = null;
int length = s.length();

// check all possible sub strings
for (int i = 0; i < length; i++) {
     for (int j = i + 1; j < length; j++) {
         int len = j - i;
         String curr = s.substring(i, j + 1);
         if (isPalindrome(curr)) {
            if (len > maxPalinLength) {
             longestPalindrome = curr;
             maxPalinLength = len;
            }
          }
    }
}

return longestPalindrome;
}


public static void isPalindrome(String s)
{
     for(int i = 0; i<s.length() - 1; i++)
     {
     if(s.charAt(i) != s.charAt(s.length() - 1 - i) {
     return false;
        }
          return true;
     }
}
```

Exercise 5 (10+4=14 Marks)

Given the following method

```
public static void mystery( int n) {
     if (n/2 == 0 ) {
        System.out.print(n);
     } else if (n % 2 == 0) {
                System.out.print("(" + n + " + ");
                mystery(n - 1);
                System.out.print(")");
            } else {
                System.out.print("(");
                mystery(n - 1);
                System.out.print(" + " + n + ")");
              }
}
```

a) What does the method display for the following call:

```
mystery(6);
```

Trace your method using a stack.

Solution:

```
mystery(6); -> (6 + ((4 + ((2 + 1) + 3)) + 5))
```

b) What does the method displays for any integer?

Solution:

The recursive method `mystery` that accepts an int `n` as a parameter and prints out the numbers 1 through `n` inclusive in a particular pattern that looks like a set of mathematical additions wrapped in parentheses. The order of the numbers should begin with all of the evens in downward order, followed by all of the odds upward from 1. Each time a number is added to the pattern, a new set of parentheses and a + sign are added too.

Exercise 6 (10 Marks)

Write a recursive method digitMatch that accepts two non-negative integers as parameters and that returns the number of digits that match between them. Two digits match if they are equal and have the same position relative to the end of the number (i.e., starting with the ones digit). In other words, the method should compare the last digits of each number, the second-to-last digits of each number, the third-to-last digits of each number, and so forth, counting how many pairs match. For example, for the call of digitMatch(1072503891, 62530841), the method would compare as follows:

```
1 0 7 2 5 0 3 8 9 1
    | | | | | | | |
    6 2 5 3 0 8 4 1
```

The method should return 4 in this case because 4 of these pairs match `(2-2, 5-5, 8-8, and 1-1)`. Below are more examples:

```
digitMatch(38, 34)          -> 1
digitMatch(5, 5552)         -> 0
digitMatch(892, 892)        -> 3
digitMatch(298892, 7892)    -> 3
digitMatch(380, 0)          -> 1
digitMatch(123456, 654321) -> 0
digitMatch(1234567, 67)     -> 2
```

Solution:

```
public static int digitMatch(int x, int y) {
      if (x < 10 || y < 10) {
         if (x % 10 == y % 10) {
           return 1;
          } else {
              return 0;
            }
      } else if (x % 10 == y % 10) {
             return 1 + digitMatch(x / 10, y / 10);
             } else {
                 return digitMatch(x / 10, y / 10);
               }
}
```

Exercise 7 (6 Marks)

Bonus

Write a recursive Java method `mergeRec` that does the same as the one implemented in Exercise 3. `mergeRec` takes two integers `x` and `y` and returns a new integer by alternating the digits of `x` and `y`. The digits should be alternated from the right to the left.

Your method should return the following for the calls:

```
mergeRec(168,734);     -> 176384
mergeRec(1,567);       -> 5617
mergeRec(1786,34);     -> 178364
mergeRec(0,0);         -> 0
mergeRec(0,123);       -> 123
mergeRec123,0);       -> 123

public static int mergeRec(int x, int y) {
```

Solution:

```
public static int f(int n, int m) {
    if (n == 0) {
      return m;
    } else if (m == 0) {
               return n;
            } else {
                    return f(n/10, m/10)*100 + (n%10)*10 + m%10;
                   }
 }
```

Scratch paper

Scratch paper

Scratch paper