German University in Cairo
Media Engineering and Technology Faculty
Prof. Dr. Slim Abdennadher
Dr. Wael Abouelsaadat
Dr. Rimon Elias

April 20, 2015

CSEN202: Introduction to Computer Programming
Spring Semester 2015

Midterm Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Please tick your major

	Major
	Civil
	BI
	Engineering

2) Duration of the exam: 2 hours (120 minutes).
3) No books or other aids are permitted for this test.
4) 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**.
5) 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**.
6) 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	$\sum$
Possible Marks	8	15	14	18	20	75
Final Marks						

Exercise 1 (8+ 2(Bonus) = 8 Marks)

The following complete program runs without error. Write the output for each `println()` statement beside that statement:

Solution:

```
public class Problem {

public static void main(String[] args) {
       System.out.println(5 * 2 - 3);

              7

       System.out.println(7 >= 6 && 5 != 3);

              true


       System.out.println(5.0 / 10);

              0.5


       System.out.println((double)(1 / 2) * 10);

              0.0


       System.out.println(10.0 / 0.0);

              Infinity


       System.out.println("100" + 10 + 1);

              "100101"


       System.out.println(0x10);

               16


       System.out.println(010);

               8


System.out.println(Integer.MAX_VALUE + 1 == Integer.MIN_VALUE);

// Integer.MAX_VALUE is the largest value that an integer can hold
// Integer.MIN_Value is the smallest value that an integer can hold
// This question is a bonus

              true
}
}
```

Exercise 2 (15 Marks)

Match each of the following code snippets with a one-line statement that would accomplish the same thing. Fill in the blanks after each code snippet with one letter from the following list. Assume that `N` is a positive integer.

Justify your answers by tracing the code with at least one example.

```
A. int c = 0;
B. int c = 1
C. int c = N/2;
D. int c = N;
E. int c = N + 1;
F. int c = N + N;
G. int c = N * N;
H. int c = (int)(Math.random() * 60);
I. int c = 10 + ((int) (Math.random() * 60));
J. int c = 10 * ((int) (Math.random() * 6));
K. int c = 10 * (1 + (int) (Math.random() * 6));
L. int c = (int) (Math.random()) * 60;
M. int c = ((int) (Math.random()) + 1) * 60;
```

a)
```
int c = 0;
for (int i = 0; i < N; i++)
     c++;
```

Accomplishes the same thing as:

Solution:

```
(d)  int c = N;
```

b)
```
int c = 0;
while (N > 1) {
   N = N - 2;
   c++;
}
```

Accomplishes the same thing as:

Solution:

```
(c)  int c = N/2;
```

c)
```
double x = Math.sqrt(N + N);
x = Math.abs(x);
int c = (int) (x * x);
```
Accomplishes the same thing as:

Solution:

```
(f)  int c = N + N;
```

d)
```
int c;
c = (int)(Math.random() * 6);
if (c < 1) c = 10;
else if (c < 2) c = 20;
else if (c < 3) c = 30;
else if (c < 4) c = 40;
else if (c < 5) c = 50;
else c = 60;
```
Accomplishes the same thing as:

Solution:

```
(k)  int c = 10 * (1 + (int) (Math.random() * 6));
```

e)
```
int c = 0;
for (int i = 0; i < N; i++) {
   for (int j = 0; j < N; j++) {
       c++;
   }
}
```
Accomplishes the same thing as:

Solution:

```
(g)  int c = N * N;
```

Exercise 3

(6+8=14 Marks)

a) Write a method that takes a number and returns the multiplication of its digits.

For example, `multiplyDigits(145)` should return `20`.

Solution:

```
public static int multiplyDigits(int x){

int result = 1;

do{
result *= x % 10;
x /= 10;

}while(x > 0);

return result;

}
```

b) Persistent numbers are numbers where the sequential product of its digits eventually produces a single digit number. For example, take the number 764.

$7 \times 6 \times 4 = 168; 1 \times 6 \times 8 = 48; 4 \times 8 = 32$; and finally, $3 \times 2 = 6$. The number of multiplication steps required to reach the single digit number from the given number is referred to as the persistence of the starting number. Thus the persistence of 764 is 4.

Write a method that takes an integer as input and returns its persistence.

Note: You have to use the method `multiplyDigits` described above with the most appropriate loop.

For example:

```
persistence(764) --> 4
persistence(2)   --> 0
persistence(23)  --> 1
```

Solution:

```
public static int persistence(int x){

int p = 0;

while(x >= 10){
x = multiplyDigits(x);
p++;
}

return p;

}
```

Exercise 4 (6+12=18 Marks)

A substring of a string `S` is another string that occurs „in“ `S`. For example `"German University"` is a substring of `"The German University in Cairo"`.

You are requested to implement a Java method that takes as input a string and returns the length of the longest substring without repeating characters.

For example, the longest substring without repeating characters for `"abcaecbb"` is `bcae`. So the length is `4`. The longest substring is not guaranteed to be unique; for example, in the string `"abcabc"`, there are several substrings with length 3 (namely `"abc"`, `"bca"`, `"cab"` and `"abc"`).

You are only allowed to use the predefined methods `charAt()` and `length()`.

a) Implement first a method `containsChar(String s, char c)` that checks whether a character `c` is in a string `s`.

For example:

```
containsChar("abcd", 'b') --> true
containsChar("abcd", 'e') --> false
```

Solution:

```
public static boolean containsChar(String s, char c) {

for(int i = 0; i < s.length(); i++)
{
if(s.charAt(i) == c)
return true;
}
return false;


}
```

b) Use the `containsChar` method to implement the method `lengthLongestSubstring`:

For example:

```
lengthLongestSubstring("aacda")  --> 3
lengthLongestSubstring("aaaaa")  --> 1
lengthLongestSubstring("aabaaa") --> 2
```

- Solution 1

```
static int lengthLongestSubString(String s){
int longest = 0;
for(int i=0;i<s.length();i++){
String temp = s.charAt(i)+"";
for(int j=i+1;j<s.length();j++){
if(!containsChar(temp,s.charAt(j)))
temp += s.charAt(j);
else
break;
}
if(temp.length()>longest)
longest = temp.length();
}
return longest;
}
```

- Solution 2

```
static int lengthLongestSubString(String s) {
String r = s.charAt(0)+"";

int max = 1;
int c = 1;

for(int j=1;j<s.length();j++){
if(!containsChar(r,s.charAt(j))){
c++;
r+=s.charAt(j);
}
else{
j-= r.length()-1;
r = s.charAt(j)+"";
if(c>=max)
max = c;
c =1;

}
}
if(c>=max)
return c;

return max;
}
```

Exercise 5 (20 Marks)

In a variation of the game of „Nim“, there are two players, you and the computer. Each takes turns picking marbles from the same pile. Whichever player is left with one marble in the pile loses the game. When picking marbles, the player must choose between 1 and $n/2$ marbles inclusive, where n is the remaining size of the pile.

For your Java program, choose a random starting pile size between 10 and 100. Assume that you are always the one who will start and the computer will just make a random choice between 1 and $n/2$. When it is the human's turn (you!), the program will ask you for your choice of marbles. If your choice is not legal, (ie. less than 1 or greater than n/2) then your program should continue to ask you, until you make a legal choice.

Here are console transcripts of one game run where you can see the initial conditions, the prompts, the display of marbles left and who wins. The first game run is a run with 21 marbles where the human player will win:

```
Starting with 21 marbles.
You play first.
Choose between 1 and 10 marble(s): 11
You entered a number outside the possible range!
Choose between 1 and 10 marble(s): 10
You chose 10 marble(s).

11 marble(s) left.
The computer chose 4 marble(s).

7 marble(s) left.
Choose between 1 and 3 marble(s): 1
You chose 1 marble(s).

6 marble(s) left.
The computer chose 2 marble(s).

4 marble(s) left.
Choose between 1 and 2 marble(s): 1
You chose 1 marble(s).

3 marble(s) left.
The computer chose 1 marble(s).

2 marble(s) left.
Choose between 1 and 1 marble(s): 1
You chose 1 marble(s).

1 marble(s) left.

You win!!
```

The second game run is a run with 10 marbles where the human player will lose:

```
Starting with 10 marbles.
You play first.
Choose between 1 and 5 marble(s): 5
You chose 5 marble(s).

5 marble(s) left.
The computer chose 2 marble(s).

3 marble(s) left.
Choose between 1 and 1 marble(s): 1
You chose 1 marble(s).
```

```
2 marble(s) left.
The computer chose 1 marble(s).

1 marble(s) left.

You lost!!
```

You are required to implement a Java program that simulates the moves of both players (You and the computer).
Note: Use the most appropriate loop to solve the problem.

Solution:

```
import java.util.*;

public class Nim {

    public static void main(String [] args) {

int marbles = (int)(Math.random()*90) +10;

      System.out.println("Starting with " +marbles + " marbles");
      System.out.println("You play first.");
      Scanner sc = new Scanner(System.in);
      int user, computer;

      do{
     System.out.println("Choose between 1 and " + marbles/2 + " marble(s)" );
        user = sc.nextInt();

        while(user<1 || user>marbles/2){
     System.out.println("You entered a number outside the possible range!");
     System.out.println("Choose between 1 and " + marbles/2 + " marble(s)" );
            user = sc.nextInt();
     }

     System.out.println("You chose "+ user + " marble(s)");

     marbles -= user;
     System.out.println(marbles + " marble(s) left.");

     if(marbles ==1){

     System.out.println("You win!!");
     break;
     }

     computer = (int)(Math.random()*(marbles/2)) +1;

     System.out.println("The computer chose " + computer + " marble(s)");

     marbles-= computer;
     System.out.println(marbles + " marble(s) left.");
     if(marbles ==1){

     System.out.println("You Lost!!");
     break;
     }


     }while(marbles >1);

    }


}
```

Scratch paper

Scratch paper

Scratch paper