Faculty of Arts & Sciences
Department of Computer Science
CMPS 200—Introduction to Computer Programming
Summer 2014–15

Programming Assignment 4

Due Tue. July 7, 11:55 pm

Problem 1 – Caesar Cipher

An important part of designing computer networks is protecting information from being read by someone other than the intended recipient. A common way to solve this problem is to encrypt a message with a key before it is transmitted so that it looks like gibberish; the recipient has a copy of the key and, as such, can decrypt the encrypted message. A simple way to encrypt information is called a Caesar cipher (because Julius Caesar was the first to use it). In a Caesar cipher, a number between 0 and 25 is selected as the key for the message. Each letter is then shifted by the selected key. For example, if the key is 1 and the original letter is 'A', then the encrypted letter will be 'B'. If the key is 3 and the original letter is 'b', then the encrypted letter will be 'e'.

Write a Java program, `Caesar`, which asks the user to enter his/her full name and outputs the encrypted version of the input. Use as key the least significant digit of your ID, so if your ID is 201001014, the key for your cipher will be 4. Two sample runs are shown below. Your program's I/O should match these sample runs (note that different keys produce different outputs). The user's input is shown in **bold**.

Sample Run 1 (key = 1):

```
C:> java CaesarCipher
Please enter your full name: Hassan Kamel Al-Sabbah
Original Name  = Hassan Kamel Al-Sabbah
Encrypted Name = Ibttbo!Lbnfm!Bm.Tbccbi
```

Sample Run 2 (key = 3):

```
C:> java CaesarCipher
Please enter your full name: Nassim Nicholas Taleb
Original Name  = Nassim Nicholas Taleb
Encrypted Name = Qdvvlp#Qlfkrodv#Wdohe
```

Problem 2 – Longest Even Run

Write a java program, `LongestEvenRun`, which reads a string of digits from the user and reports back to the user the starting index, length, and contents of the longest run of even digits within the string of digits read. In case of a tie (i.e., there are two longest sequences of even digits of the same length), your program should return information about the first sequence. In the sample runs below, the user's input appears in bold.

Sample Run 1:

```
C:> java LongestEvenRun
Enter a string of digits: 22446600
Longest run of even digits
starts at char: 0
```

```
length of run: 8
contents of run: 22446600
```

Sample Run 2:

```
C:> java LongestEvenRun
Enter a string of digits: 1133557799
String "1133557799" does not contain any even digits.
```

Sample Run 3:

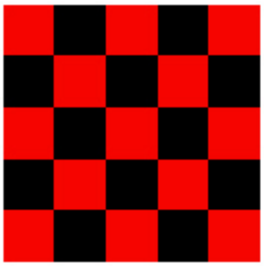
```
C:> java LongestEvenRun
Enter a string of digits: 110
Longest run of even digits
starts at char: 2
length of run: 1
contents of run: 0
```

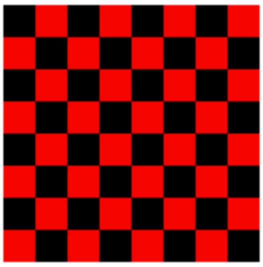
Sample Run 4:

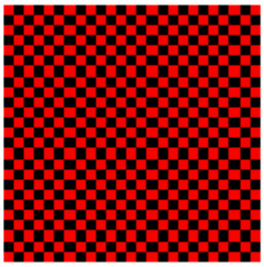
```
C:> java LongestEvenRun
Enter a string of digits: 22146878889
Longest run of even digits
starts at char: 3
length of run: 3
contents of run: 468
```

Problem 3 – CheckBoard

Write a program *CheckerBoard.java* that takes a command line input *N* and plots an *N-by-N* checkerboard. Always color the lower left square red. Draw the squares in red and black. Below is the desired output for N = 5, 10, and 25.







Write a variation on the program that reads three command line parameters and draws three corresponding checkerboards.

Problem 4 – InsertionSort

Write an `InsertionSort.java` program that reads an array of integers from standard input and sorts the array in increasing order using an insertion sort method. Insertion sort works by repeatedly looping through the array: at iteration i, the method should insert the i-th element in a location such that everything to its left is smaller that it, and "pushing" all elements between the target location and the i-th location to the right. See the Wikipedia entry http://en.wikipedia.org/wiki/Insertion_sort for an illustration.

Insertion sort is a famous sorting algorithm that has the property that after k iterations, the first k+1 elements are in sorted order. When humans manually sort something (for example, a deck of playing cards), most use a method that is similar to insertion sort.

Your program should contain the following methods:

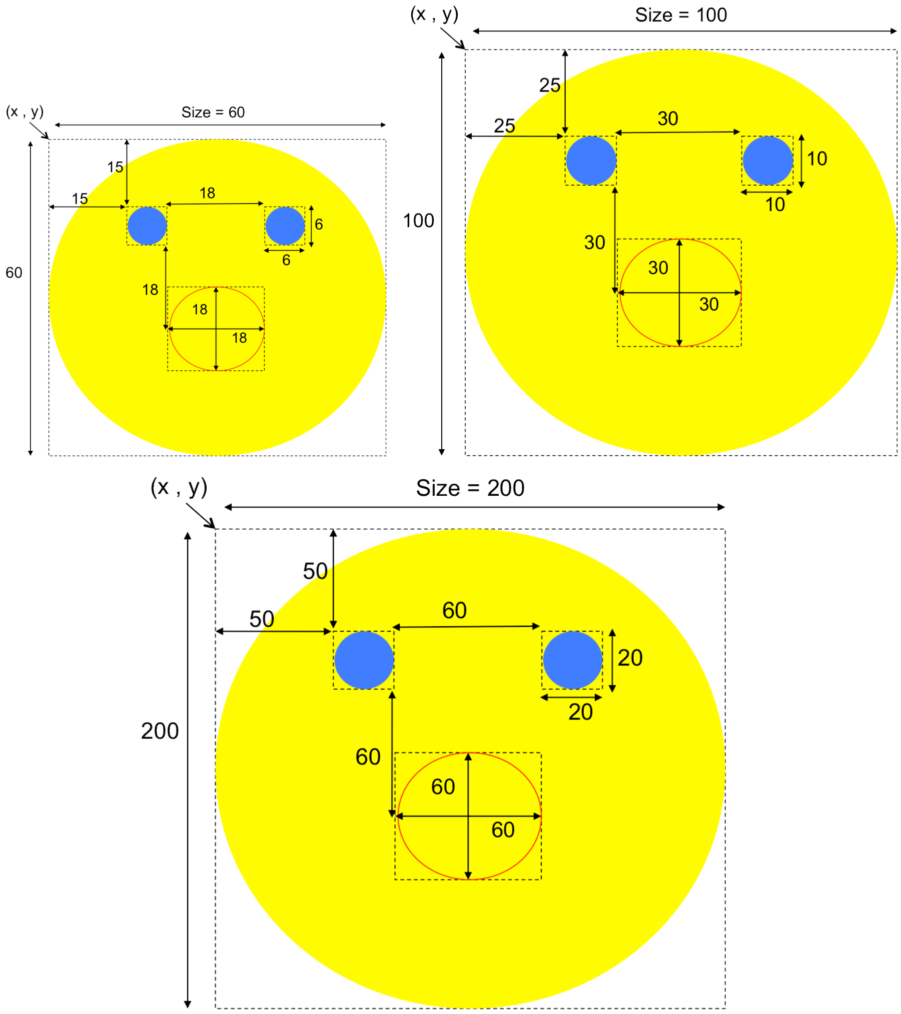
```
public static void insertionSort(int[] a)          // insertion sort an array in place
public static void printArray(int[] a)             // prints an array to standard output
```

and a main method that reads from standard input, creates an array of ints, and then calls `insertionSort()`, and `printArray()` as needed. (You may assume that the first entry in the input is the number of elements in the array and is followed by the values in the array.)

In addition, you should write a method that doesn't change its argument, but returns a new sorted array
`public static int[] insertionSort2(int[][] a)`
Hint: Use the `insertionSort()` method you wrote above to implement this method.

Problem 5 – Mood

Write a method `drawFace` that takes as arguments the *coordinates (x, y)* as integers, the integer *size,* and the string *mood,* and draws a (*Happy*, *Sad* or *Fine*) smiley face based on the following scale reference:

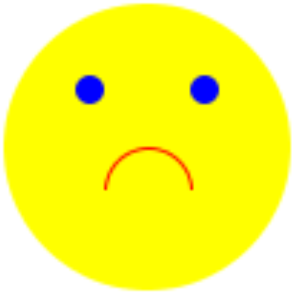


The face should be in color yellow, the eyes in color blue, and the mouth in color red. The width and height of the drawing panel could be of any size, as long as it is larger than the face's size. **The mood is case sensitive**.

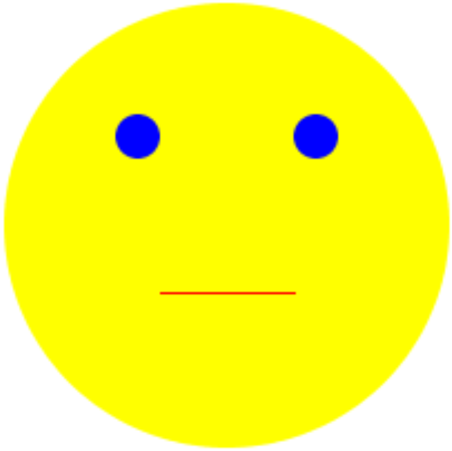
Write a program `Face.java` that uses the method `drawFace` to draw a smiley face based on the *coordinates (x, y), size* and *mood* entered by the user.

Examples:

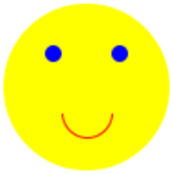
```
> java Face
Enter the x coordinate of the face: 30
Enter the y coordinate of the face: 30
Enter the size of the face: 100
Are you Happy, Sad, or Fine? Sad
```



```
> java Face
Enter the x coordinate of the face: 40
Enter the y coordinate of the face: 40
Enter the size of the face: 200
Are you Happy, Sad, or Fine? Fine
```



```
> java Face
Enter the x coordinate of the face: 30
Enter the y coordinate of the face: 30
Enter the size of the face: 60
Are you Happy, Sad, or Fine? Happy
```



```
> java Face
Enter the x coordinate of the face: 30
Enter the y coordinate of the face: 30
Enter the size of the face: 60
Are you Happy, Sad, or Fine? I hate CMPS 200
Unknown Mood!
```

Submission Instructions and Notes

- Submit your commented source code in a zip file to Moodle by the deadline.
- Your zip submissions should be named `asst4_`*`netid`*, where *`netid`* stands for your AUBnet user name. For example, if your AUBnetid is abc65, you should call your submission asst4_abc65.