

Programming Problems  
17<sup>th</sup> Annual Computer Science Programming Contest

Department of Mathematics and Computer Science  
Western Carolina University  
April 4<sup>th</sup>, 2006

## Criteria for Determining Scores

Each program should:

1. Execute without error, solve the appropriate problem efficiently and satisfy all the conditions specified in the problem statement.
2. Follow good programming style conventions such as avoiding confusing and unnecessary code.
3. Have good documentation and easily understandable variable names.
4. Start with an initial comment that includes:
  - a. The name of the team's school
  - b. The team's number
  - c. The names of the team's members

An example starting comment is:

```
// Mytown High School, Team 3, Erik Johnson, Samantha Carter, and Derek Hart
```

The score assigned each program will be based upon the extent that the program satisfies the properties listed above. The total accumulated score for all the programs represents the team score. All programs have the same weight in the scoring.

## Administrative Notes

1. Each team may use two Java reference books and 3 pages of notes.
2. Each team may also look at the Java API on the web; however, no other site on the web may be used.
3. Each team may use up to **two** computers.
4. **Inappropriate code comments will result in a lower score and are grounds for disqualification for an award.**

## Programming Notes

### General

Where appropriate, you should do input verification. In other words if the user is supposed to input, for example, a positive number, an appropriate error message should be displayed when the user does not enter a positive number. The user should then get additional chances to enter input.

### Java

The version of Java installed in the electronic classrooms here is the Java SDK 1.5.

Output with Java 1.5 uses the standard form: `System.out.println` (or `System.out.print` as appropriate)

Input with Java 1.5 input can be handled by either of the following methods:

- `BufferedReader`
- `Scanner`

Use whichever method you are most experienced with.

#### Input with `BufferedReader`:

- When using the `BufferedReader` make sure that the first statement you include in your Java program is the statement:

```
import java.io.*;
```

- Before doing any input from the keyboard you will need to declare a `BufferedReader` object that is tied to `System.in`. (In the following example the `BufferedReader` object is named "br")

```
BufferedReader br = new BufferedReader (new  
InputStreamReader (System.in));
```

- Then you may use this object to read Strings (one line at a time) from the keyboard:

```
System.out.println("Please enter a line of input> ");  
String line = br.readLine();  
System.out.println("Please enter another line> ");  
String line2 = br.readLine();
```

The above example prompts the user for input and reads two lines of input from the keyboard. `line1` references the first string, while `line2` references the second.

- Use `Integer.parseInt` or `Double.parseDouble` to convert any Strings to ints or doubles, respectively. Reading in a line of input and converting it to a single numeric type can be done in one statement. For example:

```
int i = Integer.parseInt( br.readLine() );
```

- If a line of input has NUM\_ELEMS numeric values you can use a String tokenizer to divide the line into tokens and then convert each token into the appropriate numeric value; For example:

```
import java.io.*;
import java.util.*;
...
StringTokenizer line = new StringTokenizer(" ");
try{
    line = new StringTokenizer(br.readLine());
} catch (Exception e) {}
for (i = 0; i < NUM_ELEMS; i++)
    array[i] = Integer.parseInt(line.nextToken());
```

- Finally, any methods that contain a call to `readLine()` (as well as methods that call a method that calls `readLine()` need a "throws Exception" clause in the method header, unless you want to deal with the Exception handling and place the `readLine()` call in a try...catch block. **This type of exception handling is NOT required for this contest.** So it is recommended that you just include the "throws Exception" clause.

```
public static void main(String[] args) throws Exception
{
    // throws exception code in here.
}
```

## Input with the Scanner class

- When using the Scanner class for input you will need to include the following import statement as the first line in all your Java files:

```
import java.util.Scanner;
```

- The next step is to declare a Scanner object that is tied to `System.in`

```
Scanner scanIn = new Scanner(System.in);
```

- The scanner object can now be used to read Strings, doubles and ints directly from the keyboard. The methods `nextInt()` and `nextDouble()` allow you to read integers or doubles from the keyboard; for Example:

```
System.out.println("Enter quantity of toys> ");
int number = scanIn.nextInt();
System.out.println("Enter price for each item> ");
double price = scanIn.nextDouble();
```

- The `nextLine()` method returns the next line of input from the keyboard, `nextLine()` uses the newline character to determine the end of input. The `next()` method returns the next word from the input, `next()` uses whitespace (tabs, spaces or newline characters) to determine the end of input.

```
System.out.println("Enter name> ");
String name = scanIn.nextLine();
System.out.println("Enter name> ");
String anotherName = scanIn.next();
```

- `nextLine()` would return a String consisting of multiple words such as "John Smith", while `next()` would return a single word, "John".
- The Scanner class has methods that test to see if it is possible to read a specific type of input as the next item.

```
boolean hasNext()
boolean hasNextDouble()
boolean hasNextInt()
boolean hasNextLine()
```

These functions return true if it is possible to read any non-empty String, double, int or line of input, respectively. The following example reads an undetermined number of integers from the keyboard. All these functions may BLOCK if there is NO input available. In particular, pressing the enter key will have no apparent effect as the method `hasNextInt()` is expecting some valid input besides whitespace.

```
System.out.println("Enter numbers to add (q to quit)> ");
while (scanIn.hasNextInt())
{
    num = num + scanIn.nextInt();
}
System.out.println("num is " + num);
```

This program will stop when input other than an integer is entered.

- Finally, any methods that contain a call to `next()`, `nextLine()`, `nextInt()`, or `nextDouble()` (as well as methods that call one of the above methods) need a "throws Exception" clause in the method header, unless you want to deal with the Exception handling and place the methods call in a try...catch block. **This type of exception handling is NOT required for this contest.** So it is recommended that you just include the "throws Exception" clause.

```
public static void main(String[] args) throws Exception
{
    // Scanner methods in here.
}
```

## Other Modifications with Java 1.5

### Generics

- With Java 1.5 collections are generic classes with type parameters. For example, `ArrayList<E>` collects elements of type `E`; `Map<K, V>` maps keys of type `K` to values of type `V`. When a generic type is used, the type parameters are replaced with the actual types; for example, `ArrayList<Dog>` or `Map<Location, Dog>`. In versions of Java before 1.5 collections store elements of type `Object`.

#### Java 1.5

```
private ArrayList<Dog> dogs;  
private Map<Location, Dog> environment;
```

#### Pre Java 1.5

```
private ArrayList dogs; // contains Dog objects  
private Map environment; // Maps Location objects to Dog objects
```

### Autoboxing

- Autoboxing is the automatic conversion between primitive types and corresponding wrapper classes.

```
ArrayList<Integer> numbers = new ArrayList<Integer>();  
// 25 is automatically converted to new Integer(25)  
numbers.add(25);  
int num = numbers.get(0); // intValue is automatically called
```

## Problem1: Olympic Medals!

The Olympics have just ended and several countries are clamoring to find out who won. However, we have two ways of scoring the games. The first way is called the "American" method. In this method the scoring is based on the number and type of medals a country has won. Gold medals count the most, silver is used to break ties and finally bronze medals are looked at if a country has the same number of gold and silver. Several countries (who usually score less gold medals) use the "Canadian" method of scoring instead of the "American" method. Using the "Canadian" method the total number of overall medals determines the winner of the Olympics.

You will write a program that computes the medal winner for the Olympic games. You will:

- Ask the user what scoring method they would like to use to score the Olympics.
- Prompt the user for the number of countries competing in the current Olympics.
- Read in the country names and their medal count.
- You will print out the name of the winning country.
- In case of a tie, print out the winning countries in input order.

Sample input:

```
Please enter the scoring method.
(American or Canadian) >American

Number of countries competing? > 3
Please enter country and medal count >Germany 3 2 1
Please enter country and medal count >America 2 3 1
Please enter country and medal count >Canada 1 2 4
The winner is: Germany with 3 gold medals, 2 silver medals, 1 bronze medals
```

```
Please enter the scoring method.
(American or Canadian) >Canadian

Number of countries competing? > 3
Please enter country and medal count >Germany 3 2 1
Please enter country and medal count >America 2 3 1
Please enter country and medal count >Canada 1 2 4
The winner is: Canada with 1 gold medals, 2 silver medals, 4 bronze medals
```

## Problem 2: Scrabble calculator

You are a die-hard scrabble player, a game where you make words out of tiles. Different letters of the alphabet are worth different amount of points. You play with a group that changes the point values of the letters every time they play (just to keep things interesting). To help yourself play a better game, you decided to write a program that will read in from a file a list of letters and their points values, in the form:

```
a 10
x 2
```

Each line consists of a single letter and its point value in the game. The letters do not have to be in any predefined order, and not all letters need to be listed. If a letter is not present, it has a point value of zero. You can assume that each line in the input file consists of a letter, whitespace, and an integer. If a letter is present in the file twice, it is an error. If a letter is assigned a negative value, it is also an error. If you encounter any errors in the input file, report the error and quit without scoring the word. Your program should quit after the first error discovered in the input file.

Your program should:

- Read in the input file "input.dat"
- Print out a message telling the user that the input file is valid or not.
- Prompt the user for two words they want to construct.
- Determine and print the winning word (along with its total point value)
- Uppercase and lowercase letters should be treated the same.

Sample run:

```
Input file is valid.
```

```
Please enter two words to play in scrabble...
```

```
Monkey
```

```
apple
```

```
With a score of 13 vs 12, "monkey" beats "apple".
```

Run with the input file:

```
a 3
e 5
t 7
p 2
z 5
r 6
m 3
n 1
k 4
```

Another sample run:

Input file is invalid.

Negative value -9 found for letter p.

Run with the input file:

```
a 3
e 5
p 2
z 5
p -9
m 3
n 1
k 4
z 1
```

### Problem 3: Lemming population

There is a fierce debate raging around the current population of lemmings in North Carolina. A group of students from Western Carolina University went out into the field to observe the lemmings in their natural habitat and to form opinions on the state of the lemming population. Each student formed an opinion based upon the number of lemmings they observed. Students who saw the same number of lemmings have the same opinion. If the students observed different numbers of lemmings, they will have different opinions.

You need to correlate the lemmings research data and generate a report for the NC state wildlife commission. The information was stored in a file called "opinion.dat"

The first line of the file contains the number of students that participated in the study.

Then each line after the first contains two numbers, the student id and the number of lemmings observed in the wild. **You can assume that there are no errors in the input file.**

NOTE: Students may have gone into the field multiple times, so there may be multiple entries for each student. In this case the sum of the number observed in each of these entries is the opinion for this student.

Your report should classify the different number of opinions formed about the lemmings, and which students share the same opinion.

#### Sample run:

Lemming Wildlife Study Report:

```
Student(s) 1, 5, 10 share the same opinion
Student(s) 2, 9 share the same opinion
Student(s) 3, 6 share the same opinion
Student(s) 4, 8 share the same opinion
Student(s) 7 share the same opinion
There are a total of 5 opinions
```

#### When run with the input file

```
10
1 3
2 4
3 1
4 2
5 3
6 1
7 9
8 2
9 3
10 1
9 1
10 2
```

Problem 4: You are working for a small company in northern Siberia. This company produces great software, but the operating system is a little slow. You want to write a program that when given a directory name, will list all the files and directories within the directory specified. If the directory contains a subdirectory, then all the files and directories within that subdirectory should be listed as well. Each time the files within a directory are listed they should be indented two spaces.

When the program first asks for a directory name, it should print an error message if the name input does not exist, or is not a directory! **For this program you should make use of recursion for full credit.**

Here is a sample run of the program:

```
Please enter a directory name >m ydir
Invalid directory name, please try again.
```

```
Please enter a directory name >mydir
```

```
mydir
  work
    file1
    file2
    dir1
      collect.txt
  one.dat
  listing.dat
```

In the above example, there are two files and a subdirectory within the directory "mydir": "work", which is a subdirectory, "one.dat" and "listing.dat". In the "work" directory there are two files and one subdirectory: "file1", "file2", and "dir1". The subdirectory "dir1", contains one file: "collect.txt".

Notice that each time we travel down into a new directory, the files listed are indented two spaces beyond the current indentation.

Hint:

- Using the File class, located in java.io.\*; may be useful. Here are some method prototypes:

// Constructor

◦ File(String filename) // Creates a File object for the given file or pathname

// Methods

◦ boolean exists() // Does this file denoted by this object exist?

◦ boolean isDirectory() // Is this file a directory.

◦ File[] listFiles() // Return an array of File objects in the directory represented by this object.

◦ String getName() // Get the name of this file.

- More information about the File class is available from the Java API.

Problem 5 (Tie Breaker): Your English teacher is grading term papers, but he cringes at the sight of palindromes (words or sentences that are spelled the same backwards or forwards). You decide that you will write a program that can check a sentence or word and determine if it is a palindrome, for the sake of your English grade.

You decide that since your term paper is due tomorrow, you will try to re-use your code as much as possible, and will write a recursive method called "isPalindrome", that determines whether a String is a Palindrome or not.

Your program should allow the user to enter a line of text, and print out the sentence specifying if it is a palindrome or not. Your program should ignore all whitespace, punctuation and letter case.

Thus "Madam", "mA,dam", and "madam" are all palindromes.

NOTE: Your method *isPalindrome* must be a recursive method.

Here is a sample run of the Program.

```
Enter a possible palindrome >Madam, my name is Adam!  
No "Madam, my name is Adam!" is not a palindrome.
```

```
Enter a possible palindrome >Madam, I'm Adam  
Yes "Madam, I'm Adam" is a palindrome
```