University of Windsor
SCHOOL OF COMPUTER SCIENCE

14th

Windsor Regional Secondary School Computer Programming Competition

Hosted by
The School of Computer Science, University of Windsor

WORKSHOP II

[Overview of Algorithms + Sample Competition]

Student Guide

Author:
Dr. Ziad Kobti
School of Computer Science
University of Windsor

Please email: kobti@uwindsor.ca or call 519-253-3000 ext. 2990 or 3773.

References:

Online repository of questions with automated judge:
https://uva.onlinejudge.org/

A good Java Tutorial:
http://docs.oracle.com/javase/tutorial/index.html

Guide available online:
http://kobti.myweb.cs.uwindsor.ca/sspc

Credits:

Team Coaches, Parent volunteers
Student Volunteers

Special thanks to the volunteer judge Mr. Stephen Karamatos.

I/O Mini Tutorial:

Generally, problems are in a form of multiple input cases. For example, consider this simple program where you are given a set of pairs of integers and your program is supposed to read each pair, computes its sum, prints it until it reads the pair 0 0.

Sample Input:
3 5
14 123
1 98
0 0

Consider testing your program using I/O redirection:
Step 1: store the input in a file call it A.in (where A is the problem letter and you would type in the file exactly what you would have typed as input when you run the program, including new lines and spaces).
Step 2: run your program the following way making use of the input redirection operator <
In Java: java A < A.in or in C/C++: ./a.out < A.in
Note: your program should never use File I/O as this is an access violation.

Program **Pseudocode** is typically like this:

```
1. main:
2.    Declare n1, n2 as integers
3.    Read(n1), Read (n2)
4.    while(n1 != 0 && n2 !=0)    // while it is not the end of input
5.    {
6.          Process(n1, n2)       // Process and output for this case
7.          Read(n1), Read (n2)   // read next input
8.    }
```

Tip on doing input in Java using the Scanner class:
A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.
For example, this code allows a user to read a number from System.in:
Setup:

```
      Scanner sc = new Scanner(System.in);
Read an integer:
      int x;
      x = sc.nextInt();
Read many long integers:
      while (sc.hasNextLong()) {
          long aLong = sc.nextLong();
      }
Read a String (the entire line):
      String line;
      line = sc.nextLine( );
```

The scanner can also use delimiters other than whitespace. This example reads several items in from a string:

```
String input = "1 fish 2 fish red fish blue fish";
Scanner s = new Scanner(input).useDelimiter("\\s*fish\\s*");
System.out.println(s.nextInt());
System.out.println(s.nextInt());
System.out.println(s.next());
System.out.println(s.next());
s.close();
```

prints the following output:

```
1
2
red
blue
```

The same output can be generated with this code, which uses a regular expression to parse all four tokens at once:

```
String input = "1 fish 2 fish red fish blue fish";
Scanner s = new Scanner(input);
s.findInLine("(\\d+) fish (\\d+) fish (\\w+) fish (\\w+)");
MatchResult result = s.match();
for (int i=1; i<=result.groupCount(); i++)
    System.out.println(result.group(i);
s.close();
```

The default whitespace delimiter used by a scanner is as recognized by Character.isWhitespace.
A scanning operation may block waiting for input.
The next() and hasNext() methods and their primitive-type companion methods (such as nextInt() and hasNextInt()) first skip any input that matches the delimiter pattern, and then attempt to return the next token. Both hasNext and next methods may block waiting for further input. Whether a hasNext method blocks has no connection to whether or not its associated next method will block.
The findInLine(java.lang.String), findWithinHorizon(java.lang.String, int), and skip(java.util.regex.Pattern) methods operate independently of the delimiter pattern. These methods will attempt to match the specified pattern with no regard to delimiters in the input and thus can be used in special circumstances where delimiters are not relevant. These methods may block waiting for more input.
When a scanner throws an InputMismatchException, the scanner will not pass the token that caused the exception, so that it may be retrieved or skipped via some other method.
Depending upon the type of delimiting pattern, empty tokens may be returned. For example, the pattern "\\s+" will return no empty tokens since it matches multiple instances of the delimiter. The delimiting pattern "\\s" could return empty tokens since it only passes one space at a time.

Overview of Problem Solving Steps

Objectives

- Understand what a problem is
- Discuss six problem solving steps

Types of Problems

1. Problems with Algorithmic Solutions
   - Have a clearly defined sequence of steps that would give the desired solution
     - E.g. baking a cake, adding two numbers
   - The sequence of steps or recipe for arriving at the solution is called the **algorithm**
2. Problems with Heuristic Solutions
   - Solutions emerge largely from the process of trial and error based on knowledge and experience
     - E.g., winning a tennis game or a chess game, making a speech at a ceremony
   - Many problems will require a combination of the two kinds of solution

What is a Problem?

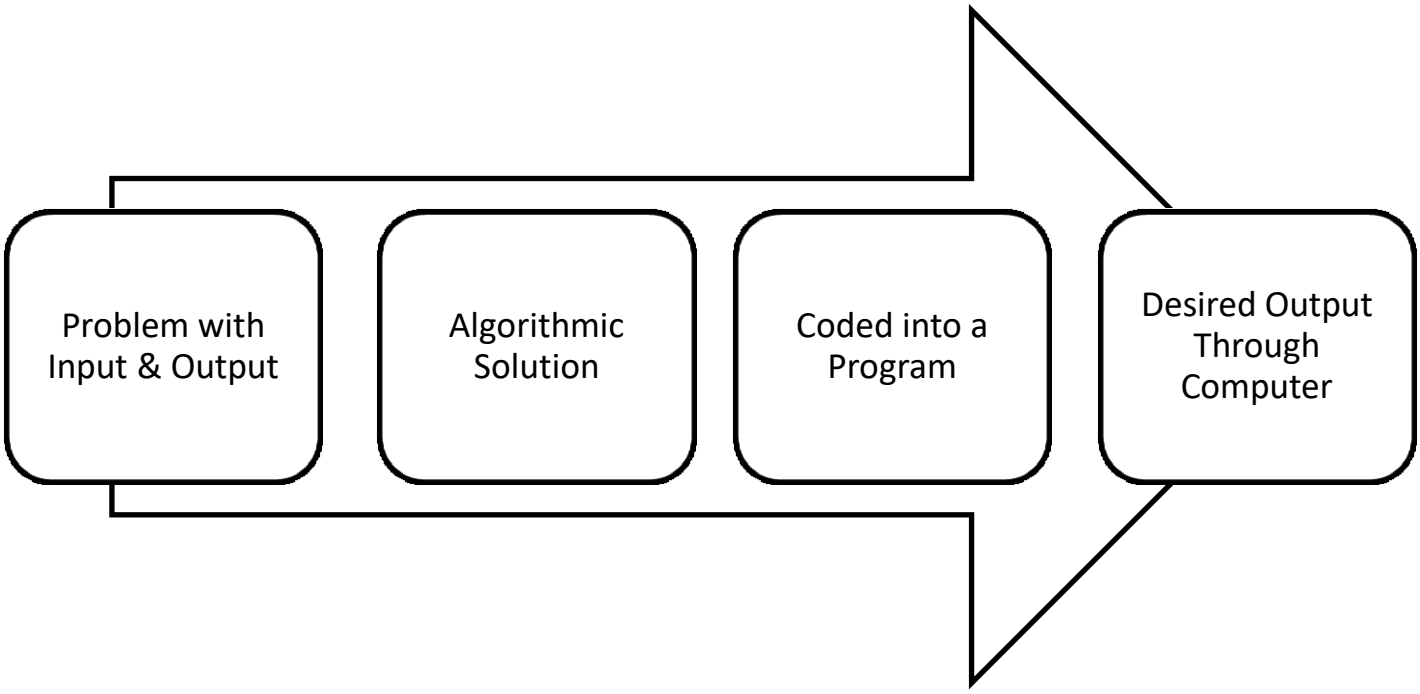
- It has some input and some desired output data, and
- we want to define a sequence of steps (algorithm and program) for transforming input data to desired output data.

What is a problem's algorithmic solution?

The sequence of steps needed to reach the desired output or the best output data expressed in **pseudocode**.

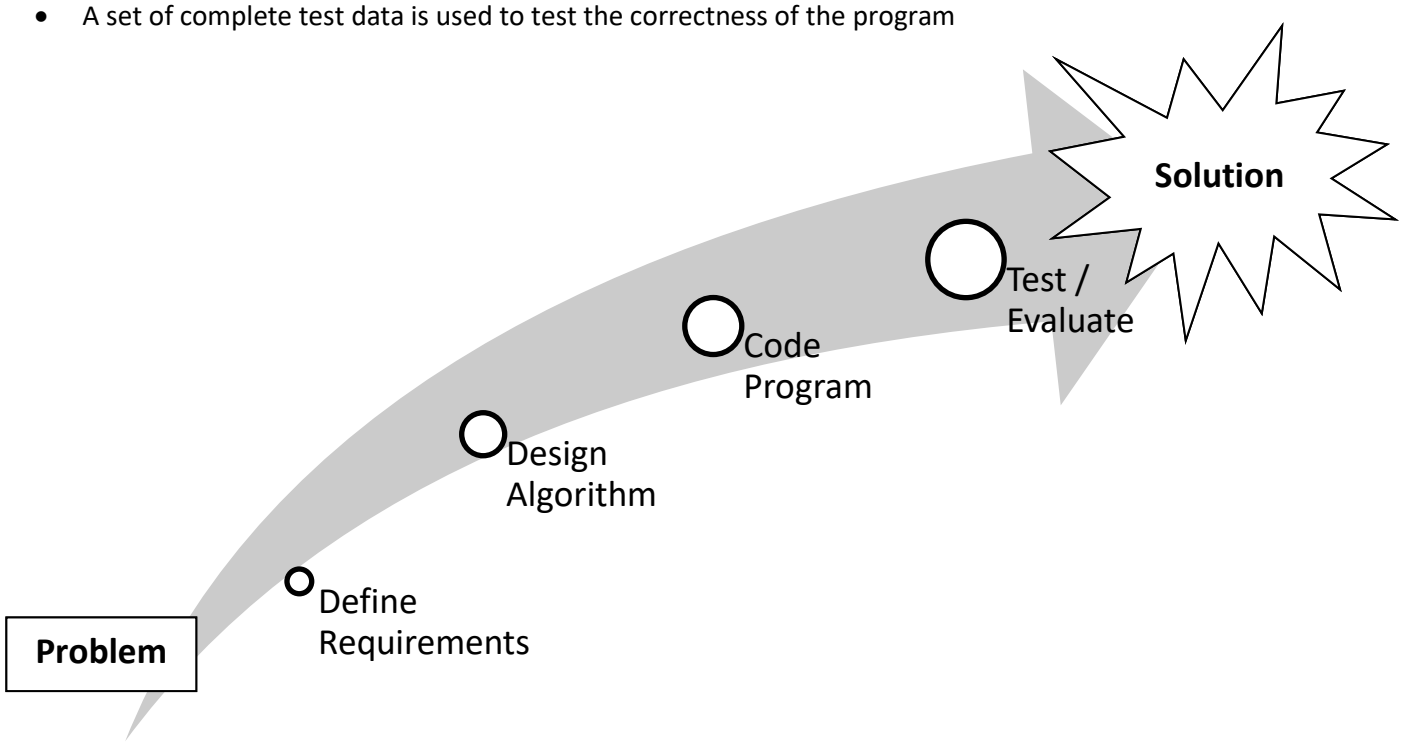
What is a Program?

The sequence of steps (algorithms) expressed (coded) in a computer language like C,C++ or Java.



Problem Solving Steps

1. Defining the Problem Requirements
    - May need knowledge or familiarity with a real life environment to understand the needs of a problem
2. Identifying Problem Components
    - Identify the problem inputs, outputs, constraints and relationships between input and output data.
3. Possibly break problem solution into small modules
    - This step uses top-down design approach to solve a big problem using structure chart. This step may be skipped for small problems.
4. Design the Algorithm to solve the problem
    - Best among many alternative ways for solving the problem is chosen.
    - Define algorithmic solution for all modules in structure chart.
    - E.g., solution that is most cost efficient, space efficient or fastest.
5. Implementation and Coding
    - Translate the algorithmic solution from step 4 to a programming language to obtain a program.
    - Programs have to obey the grammar rules (syntax) of the language and any violation results in a syntax error (called bug).
    - A bug needs to be corrected during debugging before the program is accepted by the compiler.
    - Other types of error that might need to be corrected during coding for correct results to be obtained are logic and runtime errors.
6. Evaluate the solution to ensure it produces desired results
    - A set of complete test data is used to test the correctness of the program



Difficulties With Problem Solving

- Failing to outline details of the solution (algorithm and program) completely
- Failing to define the problem correctly
- Failing to generate a sufficient list of alternatives
- Failing to use a logical sequence of steps in the solution
- Poor evaluation of the solution (algorithm and program)
- Always remember that computer does not see and needs to be given all details about what to do.

Sample Tutorials

Sample Problem A: Displaying Even Integers

Problem Statement
Given two positive integers, one on each line, you may assume that the first integer is smaller than the second. Write a program that will read the two integers from the input and then display on the screen all positive even integers that lie in the range of integers from the smaller to the larger. If one (or both) of the input integers are also even, they must be included in the output.

Sample Input:
```
2
20
```

Sample Output:
```
     2     4     6     8    10
    12    14    16    18    20
```

Sample Input:
```
13
51
```

Sample Output:
```
    14    16    18    20    22
    24    26    28    30    32
    34    36    38    40    42
    44    46    48    50
```

Note that the output format is *always* important. In this case, the values displayed *must* each be right-justified within six spaces and there must be *exactly* five values per line, except possibly for the last line.

Sample Problem A – SOLUTION in Java (problemA.java)

```
/*
 * Created on Nov 25, 2004
 *   Dr. Kobti (kobti@uwindsor.ca)
 *   University of Windsor
 *   School of Computer Science
 *
 * problemA.java
 *
 * Purpose: Reads two positive integers from a file and then displays
 *          all positive even integers between those two integers, five
 *          per line, with each integer right-justified in a field of
 *          width six spaces.  If one or both of the input integers are
 *          themselves even, they will be included in the output.
 */

import java.util.*;

public class problemA
{
      public static void main(String[] args)
      {
         Scanner sc = new Scanner(System.in);
         // Read the two integers
         int small = sc.nextInt();
         int large = sc.nextInt();

         int evenCount = 0;  // For counting the even values output

         // For each value in the given range ...
         for(int i = small; i <= large; i++)
         {
             // If the value is even ...
             if (i % 2  ==  0)
             {
                 // Display the value right-justified in 6 spaces
                 Integer I = new Integer(i);
                   for (int j = 0; j < 6-I.toString().length(); j++)
                     System.out.print(' ');
                 System.out.print(I);

                 // And count it
                 evenCount++;

                 // If 5 numbers on a line, go to next line
                 if (evenCount % 5 == 0) System.out.println();
             }
         }
          System.out.println();
      }
}
```

Sample Problem B: Sorting Names into Alphabetical Order

A list is entered and contains the names of a number of people. You do not know how many names are in the file, but you may assume that there will be no more than fifty. Each person's name appears on a separate line of the input in the following general form: Lastname Firstname
Or, to take a specific example: Gates Bill
That is, on each line the last name of the person appears first and is followed immediately by a single space, and then the first name.
Write a program that will read in all the names from the standard I/O, sort them according to last name and then display the names on the screen in that sorted order, one name per line and in the following general form: Firstname Lastname
Or, to take the same specific example: Bill Gates
For simplicity, you may assume that no two people have the same last name, and that no first or last name is longer than fifteen characters.

Sample Input:
```
Boole George
Babbage Charles
Washington George
Stroustrup Bjarne
Hollerith Herman
Turing Alan
Gates Bill
Wozniak Stephen
Jobs Stephen
Andreessen Marc
Cowpland Michael
Ritchie Dennis
Kay Alan
Pascal Blaise
Wirth Niklaus
```

Sample Output:
```
Marc Andreessen
Charles Babbage
George Boole
Michael Cowpland
Bill Gates
Herman Hollerith
Stephen Jobs
Alan Kay
Blaise Pascal
Dennis Ritchie
Bjarne Stroustrup
Alan Turing
George Washington
Niklaus Wirth
Stephen Wozniak
```

Sample Input:
```
Wirth Niklaus
Stroustrup Bjarne
Ritchie Dennis
Pascal Blaise
Gates Bill
```

Sample Output:
```
Bill Gates
Blaise Pascal
Dennis Ritchie
Bjarne Stroustrup
Niklaus Wirth
```

This problem requires knowledge of arrays, strings and at least one simple sorting algorithm. As always, output format is important. In this case the first and last names must be separated by a single space, first names must come first, and (of course) the names must be in the usual alphabetical order.

Sample Problem B – SOLUTION in Java (problemB.java)

```
/*
 * Created on Nov 25, 2004
 *   Dr. Kobti (kobti@uwindsor.ca)
 *   University of Windsor
 *   School of Computer Science
 *
 * ProblemB.java
 *
 * Purpose: Reads in a list of names containing one name per line,
 *          in the form Lastname Firstname then sorts the names
```

```
 *          according to last name and displays them one per line
 *          in the form Firstname Lastname
 */
import java.util.*;

public class problemB
{
      public static void main(String[] args)
      {
                  Scanner sc = new Scanner(System.in);

                  String[] last;
                  String[] first;
                  last = new String[50];
                  first = new String[50];
                  int numberOfNames = 0;

                  String oneLine;
                  StringTokenizer str;

                  oneLine = sc.nextLine( );
                  str = new StringTokenizer( oneLine );

              while(oneLine != null && str.hasMoreTokens())
              {
                  last[numberOfNames] = str.nextToken( );
                  first[numberOfNames] = str.nextToken( );
                  numberOfNames++;
                  oneLine = sc.nextLine( );
                  str = new StringTokenizer( oneLine );
              }

             // Sort the names using a simple "bubble" sort
                  for(int i = 1; i <= numberOfNames-1; i++)
                  {
                        for(int j = 0; j < numberOfNames-1; j++)
                        {
                              if(last[j].compareTo(last[j+1]) > 0)
                              {
                                    String temp = last[j];
                                    last[j] = last[j+1];
                                    last[j+1] = temp;
                                    temp = first[j];
                                    first[j] = first[j+1];
                                    first[j+1] = temp;
                              }
                        }
                  }

                  // Display the names in the proper format
                  for(int i = 0; i < numberOfNames; i++)
                  {
                        System.out.println(first[i] + " " + last[i]);
                  }
      }
}
```

Sample Problem Solution Sets

(For more problems refer to http://www-304.ibm.com/jct09002c/university/students/highschool/)

Average

Create a simple program that will allow the user to input 4 marks and calculate the average of those 4 marks to one decimal place.

Sample Output:

```
Enter your mark: 66
Enter your mark: 89
Enter your mark: 77
Enter your mark: 92
My average is: 81.0.
```

Solution: Average.java

```
// Average.java

import java.io.*;
class Average

{ public static void main(String[] args) throws Exception
  { InputStreamReader reader = new InputStreamReader(System. in);
    BufferedReader input = new BufferedReader (reader);

    String M;
    int mark, total=0, i=1;
    double average;

    while ( i <= 4)
    { System.out.print("Enter your mark: ");
      M = input.readLine ();
      mark =   Integer.parseInt(M);
      total += mark;
      i++;
    }
    average = total/4.0;
    System.out.println ("My average is " + average + ".");
  }
}
```

Base 4

Jerry likes to give his friends puzzles. Many of them involve some sort of code. Write a program for him that converts a base 10 number to base 4. Input and Output should both be done in the console.

Solution: Base4.java

```
// Base4.java

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Base4 {
      public static void main (String argv[]) throws IOException
      {
            int base10;
            BufferedReader reader =
            new BufferedReader( new InputStreamReader( System.in ) );


        System.out.println("Enter the base 10 number you wish to convert to base
4?");

        base10 = Integer.parseInt(reader.readLine());
        System.out.println(Integer.toString(base10,4));

      }
}
```

Reverse

Create a problem that requires a user to input a 5 digit number. Take that number and reverse it (eg. 12345, reverse=54321). Subtract the first number by the second number and produce the answer to the subtraction problem.

Example Input:

12345

Output from Example Input:

12345-54321 = 41976

Solution: Reverse.java

```
// Reverse.java

import java.io.*;

public class Reverse
{           public static void main(String[] args) throws Exception
    {   InputStreamReader r = new InputStreamReader ( System.in);
       BufferedReader input = new BufferedReader(r);

       final int u = -48;
       int a = -1;

       System.out.print ("Enter a 5 digit number: ");
       String first = input.readLine ();

       char [] m = first.toCharArray();
       int reverse[] = new int[5];

       String rn = "";
       for ( int l=0; l<5; l++)
                        reverse[4-l] = m[l];
       for ( int l=0; l<5; l++)
                        rn += reverse[l]+u;

       int fina = Integer.parseInt(first)- Integer.parseInt( rn);

       System.out.println ("Input: "+first);
       System.out.println (first+"-"+ rn+"=" + fina);

       if ( fina<0)
                        System.out.println ("Output: "+( fina*a));
       else
                        System.out.println ("Output: "+ fina);
    }

}
```

Beprisque

If a number is the only number between a prime number and a square number it is beprisque. How many positive beprisque numbers exist with less than 5 digits?

Solution: Beprisque.java

```
//Beprisque.java

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Beprisque {
      public static void main (String argv[]) throws IOException
      {
            for (int i = 1; i<10000; i++)
            {
                  if (isPrime(i-1) && Math.sqrt((double)(i+1))==
                  (int)Math.sqrt((double)(i+1)))
                  {
                        System.out.println(i);
                  }


                  if (isPrime(i+1) && Math.sqrt((double)(i-1))==

                  (int)Math.sqrt((double)(i-1)))
                  {
                        System.out.println(i);
                  }

            }

      }

      public static boolean isPrime(int num)
      {

                for (int i=2;i <=(int)Math.sqrt(num)+1; i++) {
                    if ((num%i)==0) {
                        return false;
                    }
                }

            return true;
      }
}
```

Print Pascal Triangle

Using combinations c(n,k), create a program that generates Pascal's Triangle.

Output:

```
1
1           1
1           2           1
1           3           3           1
1           4           6           4           1
1           5           10          10          5           1
1           6           15          20          15          6           1
1           7           21          35          35          21          7           1
1           8           28          56          70          56          28          8
1
```

Solution: PrintPascalTriangle.java

```
// PrintPascalTriangle.java

public class PrintPascalTriangle
{   public static void main(String[] args)
   {   for ( int i = 0; i < 9; i++)
      {    for ( int j = 0; j <= i; j++)
             System.out.print( c( i, j) + "\t");
          System.out.println ();
      }
   }

      static long c( int n, int k)
   {   if ((k <= 0) || (k >= n))
         return 1;
      return c(n-1, k) + c(n-1, k-1);
   }
}
```

Decipher

Alex sends Bob an encrypted note every Friday. Charlie has to deliver these notes to Bob, however he is very curious what they say. While delivering
the fifth note, Charlie figured out how the notes were encrypted.

The rules are:
-If there is at least one word that is a palindrome in the message, every non-space character in the message is decreased by 3 values. (e/E becomes b/B, d/D becomes a/A and c/C becomes z/Z)
-For a palindrome to count, it must have more than one letter.

-If there are no words that are palindromes in the message, reverse the order of the characters in the message.
ex. Message "abc" becomes "cba"

Note: A palindrome is a word that can be read forwards and backwards. ex. racecar

Charlie noticed that there are no special characters in the notes. Write a program for Charlie named "Decipher", that he can use to easily decipher Alex's sixth and seventh note.

Sixth Note:
esitrepxe ruoy esitrepxe ruo ekaM erom dna spuorgswen sdaolnwod erawtfos slairotut sa hcus uoy rof sgniht lufesu fo htlaew a sah latroP tnedutS MBI ehT

Seventh Note:
L fdqqrw zdlw wr wub rxw pb qhz ndbdn dw pb frwwdjh wrqljkw Grqw ohw Fkduolh nqrz ehfdxvh kh lvqw doorzhg wr frph

Solution: Decipher.java

```
//Decipher.java

import java.io.*;
import java.util.StringTokenizer;

public class Decipher
{
            StringTokenizer myTokenizer;

            public static void main (String argv[]) throws IOException
            {
                        String encMessage, decMessage;
                        Decipher d = new Decipher();

                        BufferedReader reader =
                        new BufferedReader( new InputStreamReader( System.in ) );

                        System.out.println("Enter the encrypted message");
                        encMessage = reader.readLine();

                        if (d.hasPalindromes(encMessage))

                                    decMessage = d.palDecrypt(encMessage);
                        else
                                    decMessage = d.noPalDecrypt(encMessage);
```

```
                        System.out.println("The decrypted message is:\n"+decMessage);
            }

            public String palDecrypt(String msg)
            {
                        String newMsg = "";


                        for (int i =0; i<msg.length(); i++){
                                    if (msg.charAt(i)!=' ')
                                    {
                                       if (msg.charAt(i)=='a' || msg.charAt(i)=='b'
||
                                           msg.charAt(i)=='c' ||
                                            msg.charAt(i)=='A' || msg.charAt(i)=='B'
||
                                            msg.charAt(i)=='C')

                                                {
                                                            newMsg +=
                                                            (char)(msg.charAt(i)+23);
                                                }else{
                                                            newMsg +=

                                                            (char)(msg.charAt(i)-3);
                                                }

                                    }else{
                                    newMsg += ' ';
                                    }
                        }
                        return newMsg;
            }

            public String noPalDecrypt(String msg)
            {
                        StringBuilder builder = new StringBuilder(msg);
                        return builder.reverse().toString();
            }

            public  boolean hasPalindromes (String msg)
            {
                        myTokenizer = new StringTokenizer(msg);

                 while (myTokenizer.hasMoreTokens()) {
                             if (isPalindrome(myTokenizer.nextToken()))
                                    return true;
                 }
                 return false;
            }

    public boolean isPalindrome(String word)
    {
        if (word.length()<2)
            return false;
            for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) != word.charAt(word.length()-1-i)) {
                return false;
            }
```

```
        }
        return true;
    }
}
```

Programming Environment Command Reference

You have a restricted desktop environment that is a modified version of Ubuntu Linux. You can start applications using the Applications menu which is in the top right-corner of the desktop. You do not have access to the Internet.

Command Line Window

To open a new window where you can type in Unix commands, you can:

- Right-click on the desktop (background) and choose "Open Terminal"
- From the Applications Menu, open the "System Tools" menu and choose "Terminal"

Compiling

Compile your scripts with one of the following commands. Do not use the regular compile commands since those are different than what is used for judging. The judges will use these commands to compile:

```
compilegcc progname.c
compileg++ progname.cpp
compilejava Progname.java
```

Execution

To run the a compiled C or C++ program:

```
./a.out
```

To run a Java program:

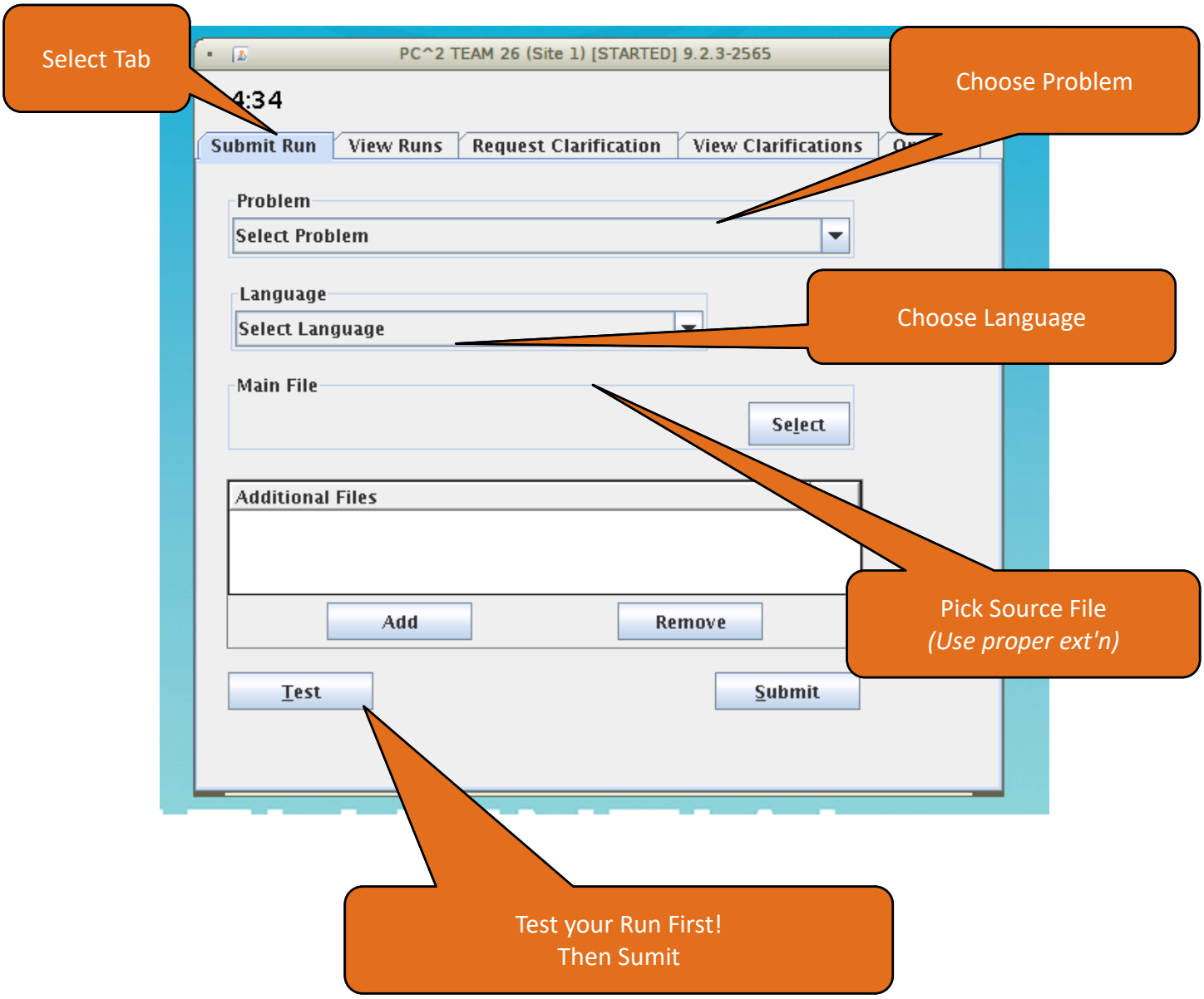
```
runjava progname
```

To run a python script, use the python2 or python3 command.

Printing

To print a file, use the following command and tell the room monitor that you have printed. Your print outs will be delivered to you in a couple of minutes.

```
printfile filename
```


Select Tab
PC^2 TEAM 26 (Site 1) [STARTED] 9.2.3-2565
4:34
Submit Run
View Runs
Request Clarification
View Clarifications
Problem
Select Problem
Choose Problem
Language
Select Language
Choose Language
Main File
Select
Additional Files
Add
Remove
Pick Source File
(Use proper ext'n)
Test
Submit
Test your Run First!
Then Sumit

Mock Competition

Rules

Competition rules are based on those used at the ACM Regional programming competitions sponsored by the ACM (Association for Computing Machinery), though there are some differences because the student level and experience are not the same.

- **Every high school in Ontario is eligible to send a team.**
- **Normally two teams per participating high school. If a high school wishes to bring additional teams for the experience, they may be registered and participate as "spares" if space permits. The "spares" do not receive any prizes or ranking scores in the final standings.**
- **There are generally three students per team though a team of two is also allowed (but at a disadvantage!).**
- **There is only one computer workstation per team, and only one login session per team is permitted.**
- **Students may use hard-copy (paper or book) reference material but not soft-copy (electronic) reference material. So no disks or CDs are permitted, not even Internet.**
- **No calculators, audio devices or video devices are permitted.**
- **New: This year we are using the official contest environment used by the ACM Regional Programming Competition called PC^2.**
- **No communication is permitted between teams or between teams and teachers/coaches, once the competition has started. Also, students are not permitted access to outside resources via a web browser or e-mail during the competition.**
- **Solutions are submitted electronically as instructed at the time of the competition, and to be accepted a submitted program must produce the right output values in the correct format, for each of the sample input data files. Program code itself is not read and not evaluated in any way. Input test data files used by the judges may (and generally will) include one or more data files that the teams have not seen.**
- **A solution which is not accepted as 'correct' will be rejected. A time penalty of 20 minutes is assessed for each rejected submission.**

The winner is determined by most problems solved, with ties broken by total time taken.

Problem G: Even it out

{G.java | G.c | G.cpp | G.py}

The input given to this program will consist of two positive integers, one on each line. You may assume that the first integer is smaller than the second. Write a program that will read in the two integers and display all positive integers that lie in the range of integers from the smaller to the larger. The output should have one integer per line. If one, or both, of the input integers are also even, they must be included in the output.

Sample Input:

```
2
20
```

Sample Output:

```
2
4
6
8
10
12
14
16
18
20
```

Sample Input:

```
13
28
```

Sample Output:

```
14
16
18
20
22
24
26
28
```

Inspiration from: http://cs.smu.ca/hspc/sample1.html (St. Mary's University HS Programming Competition)

Problems A-G are from the ACM Practice 2017 SET. The rest follow here.

Problem H: Redaction action

{H.java | H.c | H.cpp | H.py}

Redaction is a way changing the text for legal or security purposes. You are to write a program that will take a set of text and replace secret words with a set of asterisks (*) instead of the letters in the word. For example, if "Jack" was a secret word and you wanted to redact the sentence "Jack and Jill went up the hill", the desired output text would be "**** and Jill went up the hill". The number of asterisks should match the length of the original word.

The solution should not be case-sensitive on the first letter. As in the example before Jack is the secret word, but your procedure should replace "Jack" and "jack" on the output.

Your program will be given input in the following format:

- The first line will contain an integer n which will indicate the number of secret words on the list.
- The next n lines will be the secret words, one per line, all in lower case.
- The next lines (up to the end of the file) will be the text that needs to be redacted. This can be of any length or number of lines.

The output of your program should be the input minus n and the list of secret words, with characters of the secret words replaced by asterisks. No other changes should be made to the text.

Sample Input:

```
6
windsor
waterloo
minimum
maximum
average
science
I was recently accepted to the University of Windsor and to the University of
Waterloo. My average was 96 and I was offered a scholarship from Windsor for
a minimum of $2,000.
I plan to study computer science for a maximum of 4 years.
```

Sample Output:

```
I was recently accepted to the University of ******* and to the University of
********. My ******* was 96 and I was offered a scholarship from ******* for
a ******* of $2,000.
I plan to study computer ******* for a ******* of 4 years.
```

Inspiration: https://hspt.ucfprogrammingteam.org/hsptFiles/problemSet1988.pdf UCF High School Programming Tournament

Problem I: Quick stats

{I.java | I.c | I.cpp | I.py}

We want to produce some quick statistics for sets of numbers. For each set, calculate the sum, minimum and maximum values of each set.

For each set, print out the message:

Set = *n*, Sum = *s*, Minimum = *j*, Maximum = *k*.

Where n is the set number starting at 1*, s j* and *k* are the computed values. The output must look exactly as above (with *n s j* and *k*) replaced by the numbers. There must be a space after each comma and a period at the end of the line.

Each set will have a value *v* indicating the number of values in the set. The following *v* lines will be the elements of the list. All values will be integers. A value of *v* = 0 will signal the end of the data.

Sample Input:

```
5
8
6
7
9
5
2
10
10
0
```

Sample Output:

```
Set = 1, Sum = 35, Minimum = 5, Maximum = 9.
Set = 2, Sum = 20, Minimum =10, Maximum = 10.
```

Inspiration: https://hspt.ucfprogrammingteam.org/hsptFiles/problemSet1988.pdf UCF High School Programming Tournament

Problem J: Palindromes

{J.java | J.c | J.cpp | J.py} UVA 401

A regular palindrome is a string of numbers or letters that is the same forward as backward. For example, the string "ABCDEDCBA" is a palindrome because it is the same when the string is read from left to right as when the string is read from right to left.
A mirrored string is a string for which when each of the elements of the string is changed to its reverse (if it has a reverse) and the string is read backwards the result is the same as the original string. For example, the string "3AIAE" is a mirrored string because 'A' and 'I' are their own reverses, and '3' and 'E' are each others' reverses.
A mirrored palindrome is a string that meets the criteria of a regular palindrome and the criteria of a mirrored string. The string "ATOYOTA" is a mirrored palindrome because if the string is read backwards, the string is the same as the original and because if each of the characters is replaced by its reverse and the result is read backwards, the result is the same as the original string. Of course, 'A', 'T', 'O', and 'Y' are all their own reverses.

A list of all valid characters and their reverses is as follows.

Character	Reverse	Character	Reverse	Character	Reverse
A	A	M	M	Y	Y
B		N		Z	5
C		O	O	1	1
D		P		2	S
E	3	Q		3	E
F		R		4	
G		S	2	5	Z
H	H	T	T	6	
I	I	U	U	7	
J	L	V	V	8	8
K		W	W	9	
L	J	X	X		

Note that '0' (zero) and 'O' (the letter) are considered the same character and therefore ONLY the letter 'O' is a valid character.

Input
Input consists of strings (one per line) each of which will consist of one to twenty valid characters.
There will be no invalid characters in any of the strings. Your program should read to the end of file.

Output
For each input string, you should print the string starting in column 1 immediately followed by exactly one of the following strings.

STRING	CRITERIA
' -- is not a palindrome.'	if the string is not a palindrome and is not a mirrored string
' -- is a regular palindrome.'	if the string is a palindrome and is not a mirrored string
' -- is a mirrored string.'	if the string is not a palindrome and is a mirrored string
' -- is a mirrored palindrome.'	if the string is a palindrome and is a mirrored string

Note that the output line is to include the '-'s and spacing exactly as shown in the table above and demonstrated in the Sample Output below. In addition, after each output line, you must print an empty line.

Sample Input

```
NOTAPALINDROME
ISAPALINILAPASI
2A3MEAS
ATOYOTA
```

Sample Output

```
NOTAPALINDROME -- is not a palindrome.

ISAPALINILAPASI -- is a regular palindrome.

2A3MEAS -- is a mirrored string.

ATOYOTA -- is a mirrored palindrome.
```

Problem K: Counterfeit Dollar

{K.java | K.c | K.cpp | K.py} UVA 608

Sally Jones has a dozen Voyageur silver dollars. However, only eleven of the coins are true silver dollars; one coin is counterfeit even though its color and size make it indistinguishable from the real silver dollars. The counterfeit coin has a different weight from the other coins but Sally does not know if it is heavier or lighter than the real coins.
Happily, Sally has a friend who loans her a very accurate balance scale. The friend will permit Sally three weighings to find the counterfeit coin. For instance, if Sally weighs two coins against each other and the scales balance then she knows these two coins are true. Now if Sally weighs one of the true coins against a third coin and the scales do not balance then Sally knows the third coin is counterfeit and she can tell whether it is light or heavy depending on whether the balance on which it is placed goes up or down, respectively.
By choosing her weighings carefully, Sally is able to ensure that she will find the counterfeit coin with exactly three weighings.

Input
The first line of input is an integer *n* (*n* > 0) specifying the number of cases to follow. Each case consists of three lines of input, one for each weighing. Sally has identified each of the coins with the letters `A–L`. Information on a weighing will be given by two strings of letters and then one of the words "**up**", "**down**", or "**even**". The first string of letters will represent the coins on the left balance; the second string, the coins on the right balance. (Sally will always place the same number of coins on the right balance as on the left balance.) The word in the third position will tell whether the right side of the balance goes up, down, or remains even.

Output
For each case, the output will identify the counterfeit coin by its letter and tell whether it is heavy or light. The solution will always be uniquely determined.

Sample Input

```
1
ABCD EFGH even
ABCI EFJK up
ABIJ EFGH even
```

Sample Output

```
K is the counterfeit coin and it is light.
```