University of Illinois at Urbana-Champaign
Department of Computer Science

First Examination SOLUTION

CS 125 Introduction to Computer Science
90 minutes permitted

NOTE: Conflict exam has different questions with different responses but at the same level of difficulty.

Problem	Points	Score	Grader
1	12		
2	10		
3	14		
4	15		
5	12		
6	12		
Total	75		

1. Type Analysis – 12 points (3 points each)
For each of the code snippets below, there is **zero**, **one** or **more** type errors that the compiler would catch. If there is a type error circle "YES" and explain what the type error(s) are. Indicate what types and operators are involved in the erroneous part of the expression. Circle "No" if there is no type error. Examples:

Example	Response
`int a=4+"hi";`	`Yes:` A string value is being assigned to an integer variable.
`true==3`	`Yes:` An integer value is being compared with a boolean.
`int c=6*3;`	`No.`

(a)
```
double[] data = new double[10];
int result = (data[9] + 4) / 2.0;
```

Circle the correct response (YES or NO). If "YES" then specify the types and the operation involved that cause the type error.

Type Error(s)?	If yes, explain:
YES	CANNOT ASSIGN A DOUBLE TO AN INT (data[9] is already a double; the whole expression on the RHS will be a double type)

(b)
```
double t = 1 + TextIO.getlnInt();
boolean result = (t != true) || (t > 5);
```

Type Error(s)?	If yes, explain:
YES	CANNOT COMPARE A DOUBLE TO A BOOLEAN

(c)
```
String text = TextIO.getln();
int pos = text.length()/2;
int offset = (text.charAt(pos) - 'A');
```
(char-char gives an int)

Type Error(s)?	If yes, explain:
NO	

(d)
```
String text = TextIO.getln();
int mid = text.length()/2;
boolean x = mid > text.indexOf("::");
```
(int>int gives a boolean)

Type Error(s)?	If yes, explain:
NO	

2. Execution Analysis – 10 points

(a) Read the code below then answer the four questions on the right.

```
int a = 1, b = 2, c = 3, d=0;
boolean e = true;
boolean f = false;
while(a<c && !f) {
  c = TextIO.getlnInt();
  d ++;
  e = (2*a == c + b);
  b = c;
  if(d>10 || e) f = true;
}
TextIO.putln(d);
```

What is the smallest value that can be printed? 1

Choose one variable ('a' to 'f') that best illustrates the following concept.

One-way flag	*f*
Follower/Previous value	*b*
Stepper/Iteration counter	*d*

(b)
Add Java code below to print the value of the first element of A multiplied by the last element of A.

```
    public static void main(String[] args) {
        int A[] = new int[] {10,20,30, ... more values not shown};

        TextIO.put( A[0] * A[ A.length -1 ] );

    }
```

(c)
Carefully analyze the following code. What will it print when executed? For partial credit show your working.

```
    for (int a = 0; a < 5; a++)
       for (int b = a; b > 0; b--)
          TextIO.put(a);
```

Output: *1223334444*

(d)
Write one value for x that will cause y to be incremented in the code below.

Your answer: x= *-1 (any -ve odd integer)*

```
    int x = …not shown…;
    int y =0;
    if( (x%2 != 0) && (x<0) && (y++ < 10)) TextIO.putln();
    // y should now be one.
```

3. Conditional Cheesy Poofs – 14 Points

Foods4U Corp recommends food based on the user's cravings. Complete the program below to print only "B", "C", "P", or "S" (Bacon, Candy, Poofs & Spaghetti respectively) using the following rules:

- If none of the other rules are satisfied, then cheesy-***P***oofs are recommended.
- Cravings for Protein or Salt get ***B***acon Strips if they have more than $5 to spend.
- Cravings for Carbs get either ***C***andy if they have $1, $2 or $3,
  or ***S***paghetti if they have more than $3 to spend.

Read the following carefully: Complete the given code below. Use only if-else statements and **four** output statements – one to print each output. That is, **do not** print out "B" in two different places in the code, or assign it to a variable – your boolean expressions should be such that you do not have multiple statements in your code which print out the same letter. Do not create any additional variables or methods.

```
public class Foods4UWizard {
  public static void main (String[] args){
    TextIO.putln("Do you crave Carbs, Salt or Protein?");
    TextIO.putln("Type C, S, or P and press return");
    char crave = TextIO.getlnChar();  // assume either C, S, or P
    TextIO.putln("Dollars you can spend on this food?");
    int dollars = TextIO.getlnInt();  // assume a valid integer

if((crave=='P' || crave == 'S') && dollars >5) TextIO.put("B");
else if(crave=='C' && dollars > 0 && dollars<4) TextIO.put("C");
else if (crave =='C' && dollars>3) TextIO.put("S");
else TextIO.put("P");
```

// Please continue over-leaf or on spare page if you need more space.

```
  } // end of main method
} // end of class
```

4. Text Processing (There is no spoon) – 15 Points

Complete the following program to process only the first 64 lines from a text file, which is far longer. Use TextIO.getln() to read the next line from the text file. Your program will determine if there are any empty lines (i.e. zero length) *before* a line *starting* with "SPOO", in upper-, lower- or mixed case. There may be other characters on the same line after spoo. Your program will also determine the length of the longest line read. After processing is complete, print the result in the following exact format shown below. Do not print anything else.

```
MAX=81
FOUNDEMPTYBEFORE=yesno
```

Where *81* is replaced by the length of the longest line and *yesno* is either the word YES or NO. Print "YES" if at least one empty line was found before "spoo..." line (upper/lower/mixed case eg. SpOo), "NO" in all other cases (e.g. there is no "spoo" line).

The following string methods may be useful: `.equals(string)`, `.length()`, `.charAt(int)`, `.toCharArray()`, `.indexOf(string)`, `.toUpperCase()`.

```
   public class  {
     public static void main(String[] args) {
       TextIO.readFile("spoon.txt");

        boolean empty = false;
        String ans = "NO";
        int max = 0;
        int line = 1;
        while(line < 64) {
          String text = TextIO.getln();
          if(text.length() ==0) empty = true; // One way flag
          // if the line is not empty we'll check to see if it starts with SPOO.
          // Only set ans to Yes if we've already seen an empty line.
          else if (text.toUpperCase().indexOf("SPOO") ==0 && empty)
               ans = "YES";
          line ++;
          if(text.length() > max) max = text.length();
        } // loop
        TextIO.putln("MAX="+max);
        TextIO.putln("FOUNDEMPTYBEFORE="+answer);
} // end of main method
} // end of class
```

5. Party Manager (and I know I'm on the invite list) – 12 Points

```
  public class PartyInvite {
   public static void main(String[] args) {
    TextIO.putln("Number of people?");
    int len = TextIO.getlnInt();

A:  String[] names = new String[len];
    boolean[] invited = new boolean[len];

    int total =0;
B:  for(int i=0;  i<len;i++) {

       TextIO.putln("Name & Invitation for person "+(i+1)+"?");
       names[i]= TextIO.getln();
       invited[i]= TextIO.getln().equals("Y");
    C: if(names[i].equals("LA")) invited[i] = true;

    D: if(invited[i]) total ++;

    } // for loop ends here

    TextIO.putln("Total invites:" + total);
    TextIO.putln("Do not allow in:");
E: for(int i=0;  i<len;i++)
    if( ! invited[i] )
      TextIO.putln(names[i]);

  } // main
 } // class
```

(a) Write code in response area *A* to respectively set *names* and *invited* variables to new String and boolean arrays of length *len*.

(b) Complete the for-loop at response area *B* to read in all *len* people.

Note C and D are inside the for-loop.
(c) Add additional code at C so that if the most recently entered name equals your nickname then you are *always* invited (i.e. your boolean entry is true even if user entered "N").

(d). Write code in response area D such that the variable *total* will be equal the total number of invited guests (including yourself).

(e) Write code in response area E to print out the names of unwanted (uninvited) people. Print one name per line. Do not print anything else.

6. Cipher – 12 points

You want to decode a message from your secret admirer using a modified Caesar cipher. Your admirer usually mapped every letter of the alphabet to another letter of the alphabet a fixed distance away (i.e., for shift 3 they changed A to D, B to E, Z to C, etc.) but this time they are more cunning. The message **contains only capital letters** A,B,C…Z or **spaces** (there are no other character types). Space always decodes to a space.

Complete the program below: Take a line of text as input from the user (assume it only includes characters A...Z and spaces) and then print the decoded message. The shift is initially 3, thus 'D' should decode to 'A' and 'A' to 'X' etc. *But* each time you come across a space character, increase the shift distance by one. Assume there are less than 22 spaces in the message. Example: if the user enters "`FFF FF F`" Your program will print "`CCC BB A`".

```
public class MySecretAdmirer {
 public static void main(String[] args) {
TextIO.putln("Please enter the secret (all caps & spaces)");
String secret = TextIO.getln();

  int shift = 3;
  for(int i=0;i< secret.length(); i++) {
    char c = secret.charAt(i);
    if(c ==' ') shift ++;
    else c =  (char)('A' +  ((c-'A' + 26 - shift ) % 26));
    TextIO.put(c);
 }
} // end of main method
} // end of class  LAST QUESTION.
```