PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER EXAMINATIONS

CSC 207H/B07H
Duration — 3 hours

Examination Aids: Any handwritten or printed materials. No electronic aids.

Student Number: ⌊ ⌋

Last (Family) Name(s): _____

First (Given) Name(s): _____

Campus: (circle one) UTM UTSC StG

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This final examination consists of 6 questions on 16 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.*

Note: you may abbreviate `System.out.println` as `S.o.p`.

MARKING GUIDE

1: _____/10

2: _____/10

3: _____/15

4: _____/15

5: _____/30

6: _____/10

Good Luck!

TOTAL: _____/90

Question 1. [10 MARKS]

Assume that a directory contains the following files:

A.java

```
public class A {
  public static void main(String[] args) {
    System.out.print("Aunt Annie's");
    System.out.println(" Alligator");
  }
}
```

B.java

```
public class B {
  public static void main(String[] args) {
    System.out.print("Barber, Baby, Bubbles");
    System.out.println(" and a Bumblebee.");
  }
}
```

C.java

```
public class C {
  public static void main(String[] args) {
    System.out.print("Camel on the ");
    System.out.println("Ceiling.");
  }
}
```

D.java

```
public class D {
  public static void main(String[] args) {
    System.out.print("Dreamed a Dozen ");
    System.out.println("Donuts.");
  }
}
```

E.txt

```
Ear, Egg, Elephant E e E
```

Makefile

```
JR= @java -ea

one:    B.class
    ${JR} B

two:    C.class
    ${JR} C > Seuss.txt

three:  four five

four:   D.class
    ${JR} D

five:
    @wc E.txt

clean:
    @rm -rf *.class
    rm -rf Seuss.txt

%.class: %.java
    javac -source 1.4 $<

# space to add your own targets:
```

Part (a) [2 MARKS]

Assume you have a program `Seussify.java` that reads some text from standard input and creates a poetic variation of the input text. It prints the output the standard out. To the Makefile above, add a target called `six`, which runs `Seussify` on the input file `boring.txt` and saves the resulting poem in `funny.txt`.

Part (b) [1 MARK] Assume you have a program `Alphabet.java` that uses classes `A` and `B`. Add statement(s) to the Makefile to encode these dependencies.

Part (c) [7 MARKS]

Explain which files are created, modified, and/or deleted, and what (if anything) is printed to the screen, when the commands given below are run in the order shown.

- `make clean`

Created	Modified	Deleted	Output:

- `make`

Created	Modified	Deleted	Output:

- `make one`

Created	Modified	Deleted	Output:

- `make two`

Created	Modified	Deleted	Output:

- `make three`

Created	Modified	Deleted	Output:

- `make four`

Created	Modified	Deleted	Output:

- `make all`

Created	Modified	Deleted	Output:

Question 2. [10 MARKS]

Draw the finite state diagram that describes an employee record that meets the following specification:

- Some lines have keywords: keywords are in capital letters and appear at the beginning of a line, followed by whitespace and a string.
- A record starts with a line containing the keyword NAME.
- The next line may be a line with the keyword SPECIALTIES, followed by any number of lines describing the specialties. (There need not be a SPECIALTIES line; but if there is one, it must follow the NAME line.)
- Following the optional SPECIALTIES section, there must be two more pieces of information: employment type and weekend work, in either order. The information is written as follows:
  - The employment type is indicated by exactly one line containing one of the following three keywords: HOURLY, WEEKLY, or SALARIED.
  - Weekend work is one or two lines: one line is either of the weekend days SATURDAY or SUNDAY; if there is a second line it must be the other weekend day.
- The record ends with a line with the keyword DEPARTMENT

Do not show transitions to error states. We will assume that from any state, any input that does not match an outgoing transition leads to an error state.

Question 3. [15 MARKS]

For this question you will write a Python program that reads usernames from standard input and prints to standard output a list of groups for E5 repositories.

Usernames: the usernames do not follow any U of T conventions. Instead, they must begin with a lowercase letter, followed by 1-2 digits, then 3 letters, then a lowercase character, a dot, and the same penultimate lowercase character again. For example, `c23bbbk.k` and `b9JrZa.a` are valid usernames.

Input: input consists of two usernames per line. They are separated by whitespace and may have leading and/or trailing whitespace.

Output: the output should have the following format: each line should contain the group name and then the two usernames, separated by single tabs. There should be no leading or trailing whitespace. Use group names `cB07h0` ... `cB07h99`. If there are more than 100 groups in the class, stop processing after 100 groups and print an error message to standard error.

Error checking: if a username does not have the required format, a warning message should be displayed to standard output and processing should continue.

Complicating bits: your program should also accept a `-c` option that takes as an argument the name of a file containing a list of usernames, one per line. These usernames represent the complete class list. When run with the `-c` option, your program should create the pairs list as above but also add this feature:

- For each username, your program should confirm that the username is in the class list. If not, a warning message should be sent to standard error. The username should still be used for the groups, and processing should continue.

Question 3, continued

Question 4. [15 MARKS]

Class `ClassCheck` below has a `main` method that takes the names of two classes as command-line arguments. (These names represent classes, not interfaces or primitive types.)

Complete method `main` so that it calls each of the other static methods on the classes specified from the command-line arguments.

```
import java.lang.reflect.*;
public class ClassCheck {
  /**
   * Return true if a is a subclass of b, either directly or indirectly; false otherwise.
   * @param a the potential subclass
   * @param b the potential superclass
   */
  public static boolean isASubclass(Class a, Class b) {









  }

  /**
   * Return true if a has a public member field of exactly type b; false otherwise.
   * @param a the class to be examined
   * @param b a potential field of a
   */
  public static boolean hasAMember(Class a, Class b) {









  }
```

```
  /**
   * Return true if a has a public method whose return type is exactly class b.
   * @param a the class to be examined
   * @param b a potential return type of one of a's public methods
   */
  public static boolean returnsType(Class a, Class b) {









  }

  // complete this method to call each of your other methods once
  public static void main(String[] args) throws ClassNotFoundException {









  }
}
```

Question 5. [30 MARKS]

Consider these binary tree node definitions, which apply to all parts of this question. You get to choose the language you want to use (Python or Java). You must use the same language throughout this question.

```
# A node in a binary tree
class BTNode:

  def __init__(self, data):
    self.data = data
    self.left = None
    self.right = None
```

```
/** A node in a binary tree. */
public class BTNode {
  public Object data;
  public BTNode left;
  public BTNode right;

  public BTNode(Object d) {
    data = d;
  }
}
```

Part (a) [15 MARKS]

Using the Visitor pattern, write these two classes, both of which print to standard output:

1. A class `CountVisitor` that prints the number of nodes in the binary tree supplied to its constructor.
2. A class `SumVisitor` that works on binary trees that have only integer objects in them, and after visiting prints the sum of the values in the tree. If any non-integer objects are in the tree, print to standard error `"This tree contains non-integers"`. The root of the tree is given as an argument to the `SumVisitor` constructor.

Also write any extra classes that you need to do this properly.

Question 5 (a), continued

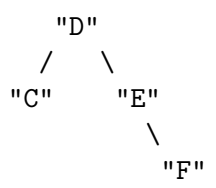
Part (b) [15 MARKS] *You might find this question particularly challenging.*

Using DOM, write a program that reads an XML file that defines a binary tree of strings as a DOM tree, converts it to a tree of `BTNode`s, and then applies `CountVisitor` to that tree. The root element of the XML file is `tree`, which contains one top-level `node` element. Each `node` element has its string specified in an attribute called `value`, and optional `left` and `right` elements that contain `node`s.

For example, you would use this XML:

```
<tree>
  <node value="D">
    <left>
      <node value="C"/>
    </left>
    <right>
      <node value="E">
        <right>
          <node value="F"/>
        </right>
      </node>
    </right>
  </node>
</tree>
```

To specify this tree:



Hint: draw the DOM tree for this example.

Question 5 (b), continued

Question 6. [10 MARKS]

Roman numerals are made using combinations of these seven letters: `I`, `V`, `X`, `L`, `C`, `D`, and `M`. Each letter has a value: `I = 1` `V = 5` `X = 10` `L = 50` `C = 100` `D = 500` `M = 1000`

Roman numerals are written with the most significant values to the left, just like regular numbers.

The digits are additive. Examples: `I` is 1, `II` is 2, and `III` is 3. `VI` is 6, `VII` is 7, and `VIII` is 8.

`I`, `X`, and `C` each be repeated up to three times, and `M` can be repeated up to 4 times. Example: `MMMCCCXII` is 3312.

The "fives" characters (`V`, `L`, `D`) are never repeated. Example: 10 is written as `X`, not `VV`.

At "nines" (9, 90, 900), you need to "subtract" from a tens character, by putting the next-lower-valued tens character before the higher-valued one. `IX` is 9, `XC` is 90, and `CM` is 900.

At "fours" (4, 40, 400) you need to "subtract" from a fives character, by putting the next-lower-valued character before a fives character. `IV` is 4, `XL` is 40, `CD` is 400.

The largest number that can be represented by this system is 4999: `MMMMCMXCIX`.

In summary, a Roman numeral has this format, in order:

1. **Thousands:** 0 to 4 `M`'s
2. **Hundreds:** Exactly one of the following:
   (a) the Roman numeral for 900
   (b) the Roman numeral for 400
   (c) an optional `D`, followed by 0 to 3 `C` characters
3. **Tens:** Exactly one of the following:
   (a) the Roman numeral for 90
   (b) the Roman numeral for 40
   (c) an optional `L`, followed by 0 to 3 `X` characters
4. **Ones:** Exactly one of the following:
   (a) the Roman numeral for 4
   (b) the Roman numeral for 9
   (c) an optional `V`, followed by 0 to 3 `I` characters

Part (a) [8 MARKS]

Write a regular expression that matches Roman numerals and allows you to extract each of the thousands, hundreds, tens, and ones as groups. It should not match anything that doesn't exactly follow the format.

Part (b) [2 MARKS]

The hundreds will be group number _____ and the ones will be group number _____.

marking template

*[This is a "blank" page. Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

*[This is a "blank" page. Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

Total Marks = 90