PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
FACULTY OF ARTS AND SCIENCE

AUGUST 2016 EXAMINATIONS

CSC 207H
DURATION — 3 HOURS
NO AIDS ALLOWED

STUDENT NUMBER: |_|_|_|_|_|_|_|_|_|_|

LAST/FAMILY NAME: _____

FIRST/GIVEN NAME: _____

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.)

This test consists of 7 questions on 24 pages (including this one).
You must receive 40% on this examination to pass the course.
When you receive the signal to start, please make sure that your copy of the test is complete.

Good Luck!

THE MATERIAL IN THIS PAGE HAS BEEN PROVIDED FOR YOUR REFERENCE

```
class Throwable:
    // the superclass of all Errors and Exceptions
     StackTraceElement[] getStackTrace() // returns the stack trace info
class Exception extends Throwable:
     Exception(String m) // constructs a new Exception with detail message m
class RuntimeException extends Exception:
    // The superclass of exceptions that don't have to be declared to be thrown
class Object:
    String toString() // returns a String representation
    boolean equals(Object o) // returns true iff "this is o"
    int hashCode() // returns the hashcode of this object
interface Comparable<T>:
    int compareTo(T o) // returns < 0 if this < o, = 0 if this is o, > 0 if this > o
interface Iterable<T>:
   // Allows an object to be the target of the "foreach" statement.
    Iterator<T> iterator()
interface Iterator<T>:
   // An iterator over a collection.
    boolean hasNext() // returns true iff the iteration has more elements
    T next() // returns the next element in the iteration
    void remove() // removes from the underlying collection the last element returned
                  // throws UnsupportedOperationException
interface Collection<E> extends Iterable<E>:
    boolean add(E e) // adds e to the Collection
    void clear() // removes all the items in this Collection
    boolean contains(Object o) // returns true iff this Collection contains o
    boolean isEmpty() // returns true iff this Collection is empty
    Iterator<E> iterator() // returns an Iterator of the items in this Collection
    boolean remove(E e) // removes e from this Collection
    int size() // returns the number of items in this Collection
    Object[] toArray() // returns an array containing all of the elements in this col
interface List<E> extends Collection<E>, Iteratable<E>:
   // An ordered Collection. Allows duplicate items.
    boolean add(E elem) // appends elem to the end
    void add(int i, E elem) // inserts elem at index i
    boolean contains(Object o) // returns true iff this List contains o
    E get(int i) // returns the item at index i
    int indexOf(Object o) // returns the index of the first occurrence of o, or -1 if
    boolean isEmpty() // returns true iff this List contains no elements
    E remove(int i) // removes the item at index i
    int size() // returns the number of elements in this List
class ArrayList<E> implements List<E>
interface Map<K,V>:
```

```
    // An object that maps keys to values.
    boolean containsKey(Object k) // returns true iff this Map has k as a key
    boolean containsValue(Object v) // returns true iff this Map has v as a value
    V get(Object k) // returns the value associated with k, or null if k is not a key
    boolean isEmpty() // returns true iff this Map is empty
Set<K> keySet() // returns the Set of keys of this Map
    V put(K k, V v) // adds the mapping k -> v to this Map
    V remove(Object k) // removes the key/value pair for key k from this Map
    int size() // returns the number of key/value pairs in this Map
    Collection<V> values() // returns a Collection of the values in this Map
class HashMap<K,V> implements Map<K,V>
class Pattern:
    static boolean matches(String regex, CharSequence input)
          // compiles regex and returns
          // true iff input matches it
    static Pattern compile(String regex) // compiles regex into a pattern
    Matcher matcher(CharSequence input) // creates a matcher that will match
class Matcher:
    boolean find() // returns true iff there is another subsequence of the
                          // input sequence that matches the pattern.
    String group() // returns the input subsequence matched by the previous match
    String group(int group) // returns the input subsequence captured by the given gro
                                          //during the previous match operation
    boolean matches() // attempts to match the entire region against the pattern.
                                 // input against this pattern
```

REGULAR EXPRESSIONS:

Here are some predefined character classes:

.	Any character
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]
\b	A word boundary: any change from \w to \W or \W to \w

Here are some quantifiers:

Quantifier	Meaning
X?	X, once or not at all
X*	X, zero or more times
X+	X, one or more times
X{n}	X, exactly n times
X{n,}	X, at least n times
X{n,m}	X, at least n; not more than m times

1: _____/10

2: _____/10

3: _____/11

4: _____/10

5: _____/10

6: _____/ 9

7: _____/10

TOTAL: _____/70