PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL 2012 EXAMINATIONS

CSC 108 H1S
Instructors: Campbell

Duration — 3 hours

Examination Aids: None

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Family Name(s): _____

Given Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully.*

This final examination paper consists of 11 questions on 20 pages (including this one). *When you receive the signal to start, please make sure that your copy of the final examination is complete.*
Comments and docstrings are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code.
You do not need to put `import` statements in your answers.
You may not use `break` or `continue` on this exam.
If you use any space for rough work, indicate clearly what you want marked.
Assume all input is valid unless otherwise indicated; there is no need to error-check.

1: _____/ 4
2: _____/ 4
3: _____/ 4
4: _____/ 8
5: _____/ 8
6: _____/ 4
7: _____/ 8
8: _____/12
9: _____/12
10: _____/ 8
11: _____/ 4

TOTAL: _____/76

Good Luck!

CONT'D...

Total Marks = 76

*[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.]*

Short Python function/method descriptions:

```
__builtins__:
  len(x) -> int
    Return the length of the list, tuple, dict, or string x.
  max(L) -> object
    Return the largest value in L.
  min(L) -> object
    Return the smallest value in L.
  open(name[, mode]) -> file
    Open a file.  Legal modes are "r" (read), "w" (write), and "a" (append).
  range([start], stop, [step]) -> list of ints
    Return a list containing the integers starting with start and ending with
    stop - 1 with step specifying the amount to increment (or decrement).
    If start is not specified, the list starts at 0.  If step is not specified,
    the values are incremented by 1.
  raw_input([prompt]) -> str
    Read a string from standard input.  The trailing newline is stripped.

dict:
  D[k] --> object
    Return the value associated with the key k in D.
  k in d --> bool
    Return True if k is a key in D and False otherwise.
  D.get(k) -> object
    Return D[k] if k in D, otherwise return None.
  D.keys() -> list of objects
    Return the keys of D.
  D.values() -> list of objects
    Return the values associated with the keys of D.
  D.items() -> list of (key, value) pairs
    Return the (key, value) pairs of D, as 2-tuples.

file (also called a "reader"):
  F.close() --> NoneType
    Close the file.
  F.read([size]) -> read at most size bytes, returned as a str
    If the size argument is negative or omitted, read until EOF (End
    of File) is reached.
  F.readline([size]) -> next line from the file, as a str. Retain newline.
    A non-negative size argument limits the maximum number of bytes to return (an incomplete
    line may be returned then).  Return an empty string at EOF.

float:
  float(x) -> float
    Convert a string or number to a floating point number, if possible.
int:
  int(x) -> int
    Convert a string or number to an integer, if possible.  A floating point
    argument will be truncated towards zero.
```

```
list:
  x in L --> bool
    Return True if x is in L and False otherwise.
  L.append(x) --> NoneType
    Append x to the end of the list L.
  L.index(value) -> int
    Return the lowest index of value in L.
  L.insert(index, x) --> NoneType
    Insert x at position index.
  L.pop() --> object
    Remove and return the last item from L.
  L.remove(value) --> NoneType
    Remove the first occurrence of value from L.
  L.reverse() --> NoneType
    Reverse the elements of L.
  L.sort() --> NoneType
    Sort the list L in ascending order.

str:
  x in s --> bool
    Return True if x is in s and False otherwise.
  str(x) -> str
    Convert an object into its string representation, if possible.
  S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are interpreted
    as in slice notation.
  S.find(sub[,i]) -> int
    Return the lowest index in S (starting at S[i], if i is given) where the
    string sub is found or -1 if sub does not occur in S.
  S.index(sub) -> int
    Like find but raises an exception if sub does not occur in S.
  S.isdigit() -> bool
    Return True if all characters in S are digits and False otherwise.
  S.lower() -> str
    Return a copy of the string S converted to lowercase.
  S.lstrip([chars]) -> str
    Return a copy of the string S with leading whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.replace(old, new) -> str
    Return a copy of string S with all occurrences of the string old replaced
    with the string new.
  S.rstrip([chars]) -> str
    Return a copy of the string S with trailing whitespace removed.
    If chars is given and not None, remove characters in chars instead.
  S.split([sep]) -> list of strs
    Return a list of the words in S, using string sep as the separator and
    any whitespace string if sep is not specified.
  S.strip() -> str
    Return a copy of S with leading and trailing whitespace removed.
  S.upper() -> str
    Return a copy of the string S converted to uppercase.
```

```
media:
  copy(Picture) -> Picture
    Return a copy of the Picture.
  create_picture(int, int) --> Picture
    Given a width and a height, return a Picture with that width and height.  All pixels are white.
  get_blue(Pixel) --> int
    Return the blue value of the given Pixel.
  get_color(Pixel) --> Color
    Return the Color object with the given Pixel's RGB values.
  get_green(Pixel) --> int
    Return the green value of the given Pixel.
  get_pixel(Picture, int, int) --> Pixel
    Given x and y coordinates, return the Pixel at (x, y) in the given Picture.
  get_red(Pixel) --> int
    Return the red value of the given Pixel.
  get_x(Pixel) --> int
    Return the x coordinate of the given Pixel.
  get_y(Pixel) --> int
    Return the y coordinate of the given Pixel.
  set_blue(Pixel, int) --> NoneType
    Set the blue value of the given Pixel to the given int value.
  set_color(Pixel, Color) --> NoneType
    Set the RGB values of the given Pixel to those of the given Color.
  set_green(Pixel, int) --> NoneType
      Set the green value of the given Pixel to the given int value.
  set_red(Pixel, int) --> NoneType
    Set the red value of the given Pixel to the given int value.
```