UNIVERSITY OF TORONTO

Fall 2013 Midterm #1

CSC180: Introduction to Computer Programming

Duration: 90 minutes

October 10th, 2013

Last Name: ______________________

First Name: ______________________

Student Number: ______________________

Circle your instructor: Ashraf Al Daoud Kaveh Aasaraai Shobhit Jain

Instructions:

- **Write your name on the back of the final page of this exam.**
- **Do not open this exam until you hear the signal to start.**
- Have your student ID on your desk.
- No aids permitted other than writing tools. Keep all bags and notes far from your desk before the exam begins.
- There are 6 questions on 16 pages including this page.
- Exams written with pencil will not be remarked.
- All your answers should be based on Python 3.
- Write comments where it would clarify your code. Docstrings are required only where explicitly indicated.
- If you use any space for rough work, clearly indicate the section(s) that you want to be marked.

Mark Breakdown

Q1:	/10
Q2:	/10
Q3:	/5
Q4:	/5
Q5:	/10
Q6:	/5
Total:	**/45**

Q1 (10 marks)

For parts (**a-i**), Indicate which of the statements is Yes or No:

a) `2nd` is a valid variable name in Python. Yes [✓ No]

b) The expression `"word"[1:2]` evaluates to `"o"`. [✓ Yes] No

c) The following statement returns `True`:
`"big" < "bigger" and "bigger" < "biggest"` [✓ Yes] No

d) The output of the following function for `n = 0` is `False`: Yes [✓ No]

```
def func1(n):
    if not n:
        return True
    return False
```

e) The following is a valid code in Python: Yes [✓ No]

```
s = "this is a string"
s[-1] = "G"
```

f) The code below has an error! [✓ Yes] No

```
s = "12345"
print(s[5])
```

g) In Python, every function must return a value. Yes [✓ No]

h) In Python 3, the expression `(10 / 4 == 2)` evaluates to `False` [✓ Yes] No

i) The Python statement below executes with no errors: Yes [✓ No]

```
number = int("2.5")
```

j) Write down in the box below the name of the variable defined by Python that holds the current module name:

__name__

Q2 (10 marks)

Write the output of the execution of the following code snippets in the provided space. Write ***as clearly and legibly as possible***. Marks will not be awarded to unreadable answers:

a) (4 marks)

```
def f(n):
   return 3 * n

def g(n):
   return 5 * n

def h(n):
   return 2 * n

def doto(value, func):
   if func == "f":
      return f(value)
   elif func == "g":
      return g(value)
   return h(value)

if _name_ == "_main_":
   print (doto(7, "f"))
   print (doto(7, "g"))
   print (doto(7, "h"))
```

Answer:
```
21
35
14
```

b) (3 marks)

```
def func(s):
    x = "pa"
    s_n = "" # This is an empty string (no space)
    for letter in s:
        if letter not in x:
            s_n += letter
    return s_n

if _name_ == "_main_":
    print(func("Applebee's"))
```

Answer:
"Alebee's"

c) (3 marks)

```
a = 3

def func1(a):
    a = a ** 2
    return a

def func2(a):
    return a ** 3

if _name_ == "_main_":
    print(func1(a), func2(a))
```

Answer:
9, 27

Q3 (5 marks)

In mathematics, the factorial of a non-negative integer `n` is denoted by `n!`. The factorial is the product of all positive integers less than or equal to `n`. For example:

- $0! = 1$
- $1! = 1$
- $2! = 2 \times 1 = 2$
- $3! = 3 \times 2 \times 1 = 6$
- $4! = 4 \times 3 \times 2 \times 1 = 24$

By convention, $0! = 1$. Write down a function `factorial(n)` which takes a non-negative integer `n` as parameter and returns its factorial. Your function should contain a proper docstring.

```
def factorial(n):
    ''' (int) -> int

    Return the factorial of n.
    '''

    result = 1

    while n > 0:
        result *= n
        n = n - 1

    return result
```

Q4 (5 marks)

Write down a function `pattern(n)` which takes a non-negative integer `n` as parameter and returns a string of asterisks which will give the following output when printed (no space between asterisks):

```
n = 3                                    n = 4

*                                        *
**                                       **
***                                      ***
                                         ****
```

Under `if _name_ == "_main_":` block, call function `pattern` with `n = 5` and print the result.

```
def pattern(n):
    ''' (int) -> str

    Return the asterisks pattern with n lines.
    '''

    result = ''
    for i in range(1, n + 1):
        result += "*" * i + "\n"

    return result.strip("\n")


if __name__ == '__main__':
    # Call function pattern with n = 5 and print the result

    print(pattern(5))
```

Q5 (10 marks)

Part (a) (4 marks)

Write down a function `longer(s1, s2)` which takes two strings `s1` and `s2` as parameters and returns the longer string. If both `s1` and `s2` are of same length then function `longer` should always return string `s1`. Your function should contain a proper docstring.

```
def longer(s1, s2):
    ''' (str, str) -> str

    Return the longer of s1 and s2. If s1 and s2 are of same length return s1.
    '''

    if len(s1) >= len(s2):
        return s1
    else:
        return s2
```

Part (b) (2 marks)

Write code to get strings `s1` and `s2` from user and print the value returned by function `longer(s1, s2)`.

```
s1 = input("String 1: ")
s2 = input("String 2: ")
print(longer(s1, s2))
```

Part (c) (4 marks)

Write down at least 4 distinct (non-redundant) test cases for function `longer(s1, s2)` to test if the function behaves as expected.

s1	s2	Expected output
”	”	”
”	'a'	'a'
'a'	”	'a'
'ab'	'cd'	'ab'
'abc'	'123'	'abc'
'ab c'	'abc'	'ab c'

Marking code:

- code (a):
- code (b):
- code (c):

Q6 (5 marks)

A car manufacturer is developing a new entertainment system that embeds a display into the car's dashboard. One of the features of the system is to show the name of the currently playing song on the display. However, to keep the cost down, the display is chosen to be only 8 characters wide, with 8 lines. As a result, song names longer than 8 characters cannot be displayed on one line.

You are given the task of writing a program that runs on the processor of this new entertainment system. Your program is given a song name as a string, and is expected to return the same name in one string, broken into multiple lines, using line break (\n) characters, at 8 character boundaries. If the song name is more than the screen capacity, that is 8 x 8 = 64 characters, your program must truncate the name and only return the first 64 characters of the song name along with the added line break characters. Note that your program adds extra line break characters to the string which will not be visible on the screen.

Write a function, `line_break_name(song_name)`, that gets the song name as parameter and returns the name broken into lines in one string. For example:

```
line_break_name('This is a very long name')
```

will return:

```
'This is \na very l\nong name'
```

```
def line_break_name(song_name):
    broken_name = ""
    lines = 0

    while len(song_name) > 0 and lines < 8:
        if len(song_name) > 8:
            broken_name += song_name[:8] + "\n"
            song_name = song_name[8:]
        else:
            broken_name += song_name
            song_name = ""

        lines += 1

    return broken_name
```

Use this page to continue your solution to question 6.

This page is left blank intentionally for extra work.

This page is left blank intentionally for extra work.

Short Python function/method descriptions:

```
input([prompt]) -> string
    Read a string from standard input. The prompt string, if given, is printed without
    a trailing newline before reading.

float(x) -> float
    Convert a string or number to a floating point number, if possible.

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

str(x) -> str
    Return a string representation of the object, if possible.

min(iterable) -> value
    min(a, b, c, ...) -> value

    With a single iterable argument, return its smallest item.
    With two or more arguments, return the smallest argument.

max(iterable) -> value
    max(a, b, c, ...[, key=func]) -> value

    With a single iterable argument, return its largest item.
    With two or more arguments, return the largest argument.

str:
    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.endswith(suffix[, start[, end]]) -> bool
        Return True if S ends with the specified suffix, False otherwise. With optional start,
        test S beginning at that position. With optional end, stop comparing S at that position.
        suffix can also be a tuple of strings to try.

    S.find(sub [,start [,end]]) -> int
        Return the lowest index in S where substring sub is found, such that sub is contained
        within s[start:end].  Optional arguments start and end are interpreted as in slice notation.
        Return -1 on failure.

    S.isalnum() -> bool
        Return True if all characters in S are alphanumeric and there is at least one character in S,
        False otherwise.

    S.isalpha() -> bool
        Return True if all characters in S are alphabetic and there is at least one character in S,
        False otherwise.

    S.isdigit() -> boolean
        Return True if all characters in S are digits and False otherwise.

    S.isspace() -> bool
        Return True if all characters in S are whitespace and there is at least one character in S,
        False otherwise.

    S.lower() -> string
        Return a copy of the string S converted to lowercase.

    S.replace(old, new[, count]) -> string
        Return a copy of string S with all occurrences of substring old replaced by new.
        If the optional argument count is given, only the first count occurrences are replaced.

    S.rfind(sub[, start[, end]]) -> integer
    Return the highest index in S where substring sub is found, such that sub is contained
    within S[start:end]. Return -1 if sub does not occur within S[start:end].

S.rstrip([chars]) -> string
    Return a copy of the string S with trailing whitespace removed. If chars is given and
    not None, remove characters in chars instead.

S.startswith(prefix[, start[, end]]) -> bool
    Return True if S starts with the specified prefix, False otherwise. With optional start,
    test S beginning at that position. With optional end, stop comparing S at that position.
    prefix can also be a tuple of strings to try.

S.strip([chars]) -> string
    Return a copy of the string S with leading and trailing whitespace removed. If chars is
    given and not None, remove characters in chars instead.

S.upper() -> string
    Return a copy of the string S converted to uppercase.
```

Last Name: ____________________

First Name: ____________________