Name: _____ Net ID: _____

NYU, Tandon School of Engineering
CS-1114: Introduction to Programming and Problem Solving — Fall 2017

CS-1114 – Midterm Exam

Tuesday, November 14, 2017

- You have one hour and 20 minutes.
- The exam has **TWO Parts**. The first part of the exam contains this cover page and a couple of pages for scratch work. **What you write in those pages will not be graded**, but you must hand it in with your exam.
- Write your Name and NetID at the head of each page.
- Write your answers clearly and concisely, in the spaces on the exam. Try to avoid writing near the edge of the page. **YOU MAY NOT USE THE BACKSIDE OF THE EXAM PAPERS**, as they will not be looked at. If you need extra space for an answer, use the **extra page at the end of the exam** and **mark it clearly**, so we can find it when we're grading.
- **Don't use pencils**, as they don't show well when scanned.
- This is a closed-book exam. Calculators are not allowed.
- There are 6 questions all together, with 100 points total. Note that there are longer programming problems at the end. Be sure to allow enough time for them.
- In all questions, you may assume that the user's inputs and function parameters are as expected. That is, if the program expects a positive integer, you may assume that users will enter positive integers. If a function expects a list as a parameter, you may assume that it will be called with a list as an argument.
- Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables and functions, choose most suitable control statements, define extra functions where it is needed, etc.
- No need to document your code in this exam, but you may add comments if you think they are needed for clarity.
- **You may not use any Python constructs that were not shown in class.**
- Read every question completely before answering it.
- Cell phones, and any other electronic gadgets must be turned **off**.
- Do not talk to any students during the exam. If you truly do not understand what a question is asking, you may raise your hand when one of the CS1114 instructors is in the room.

Name: _____ Net ID: _____

Scratch
(This paper will not be graded)

Name: _____ Net ID: _____

Scratch
(This paper will not be graded)

Name: _____ Net ID: _____

Question 1 (18 points)

For each of the following, show the final value of `my_var` at the end of executing the expressions, or write ERROR if there is an error.

	code fragment	**value of `my_var` at end (or ERROR)**
a.	`my_var = [1, 2, 3]` `my_var[3] = my_var[1] + 3`	

b.	`list1 = [1, 2, 3, 4]`
	`list2 = [4, 3, 2, 1]`
	`my_var = list1[ : 2] + list2[2 : ]`	

c.	`my_var  = [1, 2, 3]`
	`arg = 4`
	`my_var.append(arg)`	

d.	`my_var  = [1, 2, 3]`
	`arg = [4]`
	`my_var.append(arg)`	

e.	`s = 'abcABCabc'` `my_var = s.find('bc')`	
f.	`s = 'abcABCabc'` `my_var = s.find('Bc')`	

g.	`my_var  = 'abc'`
	`x = my_var`
	`x.upper()`	

h.	`my_var = [1, 2, 3]`
	`x = my_var`
	`x.insert(0, 4)`	

i.	`my_var  = [10, 20, 30, 40]`
	`my_var.insert(1, 2)`
	`my_var.pop(3)`	

Name: _____ Net ID: _____

Question 2 (10 points)
Given the following definitions:

```
def main():
    n = 3
    print("main msg1: n =", n)
    func1(n)
    print("main msg2: n =", n)

def func1(n):
    if (n == 1):
        print("func1 msg1: n =", n)
    else:
        print("func1 msg2: n =", n)
        func2(n - 1)
        print("func1 msg3: n =", n)

def func2(n):
    if (n == 1):
        print("func2 msg1: n =", n)
    else:
        print("func2 msg2: n =", n)
        func1(n - 1)
        print("func2 msg3: n =", n)
```

What would be printed when calling `main()`?

Output:

_____

_____

_____

_____

_____

_____

_____

_____

_____

Name: _____ Net ID: _____

Question 3 (10 points)
Given the following definitions:

```
def f(s):
    res_s = ''     # initializes an empty string
    for ch in s:
        res_s = ch + res_s
    return res_s


def main():
    s1 = "live for desserts"
    s2 = f(s1)
    ind1 = s1.find(' ')  # searches for a space
    ind2 = s2.find(' ')  # searches for a space

    res1 = s1[ : ind1]
    res2 = s1[(-1)*ind2 : ]
    print("res1 =", res1,"res2 =", res2)

    res3 = f(res1)
    res4 = f(res2)
    print("res3 =",res3, "res4 = ",res4)
```

What would be printed when calling `main()`?

Output:

_____

_____

_____

_____

_____

Name: _____ Net ID: _____

Question 4 (12 points)
Given the following definitions:

```
def main():
     lst1 = [1, 2, 3, 4]
     lst2 = lst1
     s = ""

     print("Before calling the function")
     print("lst1 =", lst1, "lst2 =", lst2, "s =", s)

     func(lst2, s)

     print("After calling the function")
     print("lst1 =", lst1, "lst2 =", lst2, "s =", s)


def func(lst, s):
     n = len(lst)
     for i in range(n):
         lst.insert(i, lst[i])
         s = s + "1"

     print("Inside the function")
     print("lst =", lst, "s =", s)
```

What would be printed when calling `main()`?

Output:

_____

_____

_____

_____

_____

_____

_____

_____

Name: _____ Net ID: _____

Question 5 (20 points)

Implement a function:

```
def insert_sorted(srt_words_lst, new_word)
```

This function is called with:

1. `srt_words_lst` – a list containing words (each of type `str`), appearing in an ascending lexicographical (alphabetical) order.
2. `new_word` – a word (of type `str`)

When called, it should add `new_word` into its sorted place in `srt_words_lst`. That is, it mutates the list, so that after the execution, it would also include `new_word`, and remain sorted.

For example, given the following `main` function:

```
def main()
    srt_words_lst = ["fun", "go", "know", "moon"]
    insert_sorted(srt_words_lst, "man")
    print(srt_words_lst)
```

Calling `main()` should give the following output:

```
["fun", "go", "know", "man", "moon"]
```

Implementation requirement:

1. Your implementation should **work in-place.** That is, it should mutate the list.
2. In this question, you are **not allowed to use the `sort()` method** of `list`.

Name: _____ Net ID: _____

```
def insert_sorted(srt_words_lst, new_word):
```

Name: _____ Net ID: _____

Question 6 (30 points)
This question has two sections. You should answer both.

Section A (20 points):
Implement a function:

```
def shuffle_list(lst)
```

Given a list `lst`, this function shuffles the elements of `lst` to form a random permutation (reordering) of them.

For example, given the following `main` function:

```
def main()
    lst = [7, 5, 0, -1, 4]
    shuffle_list(lst)
    print("First time:", lst)
    shuffle_list(lst)
    print("Second time:", lst)
```

Calling `main()` could give the following output:

```
First time: [5, -1, 4, 0, 7]
Second time: [0, -1, 7, 5, 4]
```

Implementation requirements:
1. Your implementation should **work in-place.** That is, it should mutate the list.
2. From the `random` module, you are allowed to use only the `randint(a, b)` function that returns a random integer in the range from `a` to `b` (inclusive)

Section B (10 points):
Write a `main()` function that asks the user for a positive integer n, and prints 3 random permutations of the integers $1,2, \ldots, n$.

Your program should interact with the user **exactly** as demonstrated below.

Execution example:

```
Enter a positive integer: 10
#1: 1, 4, 5, 2, 3, 9, 8, 7, 10, 6
#2: 4, 2, 1, 10, 8, 7, 9, 5, 3, 6
#3: 3, 1, 6, 4, 2, 7, 10, 9, 5, 8
```

Implementation requirements:
Your `main()` should use the `shuffle_list` function you implemented in section (A).

Name: _____ Net ID: _____

(A)

```
import random

def shuffle_list(lst):
```

Name: _____ Net ID: _____

(B)

```
def main():
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Name: _____ Net ID: _____

EXTRA PAGE IF NEEDED

Note question numbers of any questions or part of questions that you are answering here. Also write “ANSWER IS ON LAST PAGE” near the space provided for the answer.