15110 PRINCIPLES OF COMPUTING – EXAM 1A – FALL 2012

Name _____ Section _____

Andrew id _____

Directions: Answer each question neatly in the space provided. Please read each question carefully. You have 50 minutes for this exam. No electronic devices allowed. Good luck!

1	_____
2	_____
3	_____
4	_____
5	_____
6	_____
TOTAL	_____

1. History of computation

(a) [4 pts] Match each item in the left column with the most relevant item in the right column.

Jacquard's loom	6	1. Electromechanical computer
ENIAC	4	2. Enigma cipher
Moore's Law	7	3. Polynomial function
Difference Engine	3	4. Vacuum tubes
Harvard Mark I	1	5. First programmer
Grace Hopper	8	6. Hollerith tabulating machine
Ada Lovelace	5	7. Exponential function
Alan Turing	2	8. Debugging

(b) [2 pts] A byte is 8 bits, so a kilobyte is 1024 * 8 bits.

(c) [2 pts] Moore's Law says that the complexity of integrated circuit chips doubles every 18 months.

2. This problem focuses on expressions and data types.

(a) [6 pts] For each of the following Python3 expression, write down the value that would be output if the expression was evaluated in python3 -i.

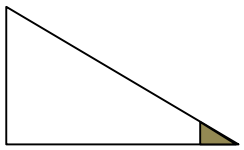
40 // 9	4	15.0 //2	7.0
2 * 2 ** 4	32	15 % 2	1
6 + 4 * 2 - 1	13	2 != 2	False

(b) [2 pts] Write a Python3 function `triangle_area` that takes two parameters `h` and `b`, respectively, for the height and base of a triangle, and returns the area of the triangle given by the formula

$$A = \tfrac{1}{2} (\text{height} \times \text{base}).$$

```
def triangle_area(h,b) :
    area = (h*b)/2
    return area
```

(c) [2 pts] Write a Python3 function `truncated_triangle` that takes height and base parameters (`h` and `b`) as input, and computes the area of a triangle with the tip cut off. (That is, the white area of the triangle shown below.) The tip is also a triangle; its height and base are 10% of the height and base, respectively, of the larger triangle, as shown in the figure. Use the `triangle_area` function in your solution.



```
def truncated_triangle(h,b) :
    h1 = h*0.1
    b1 = b*0.1
    trunc_tri = triangle_area(h,b) – triangle_area(h1,b1)
    return trunc_tri
```

(d) [2 pts]

```
def mystery1(m,n):
    i = 0
    while i <= (n-1):
      i = i + 1
      print(i ** m, end = " ")
```

The Python3 function above prints a sequence of numbers. Which of the following is the output of the function expressed in terms of m and n? Circle your answer.

1 2^m 3^m … n^m OR 1 m^2 m^3 … m^n

(e) [5 pts] If the print statement was taken outside of the while loop to occur right after the while statement, as shown below, what would the function call mystery2(2, 3) print?

```
def mystery2(m,n):
  i = 0
  while i <= (n-1):
    i = i + 1
  print(i ** m, end = " ")
```

9

(f) (5 pts) Consider the following Python3 function:

```
def mystery3(m,n):
  i = 0
  result = 0
  while i <= (n-1):
    i = i + 1
    result = result + i ** m
  return result
```

What would the value of the variable x be after executing the following assignment statement below?

```
x = mystery3(2,4)
```

30

3. This question focuses on the array data type and variable assignments.

(b) [4 pts] Assume the following list definition in Python3 using an array.

a = [1, 2, [3, 4, 5], 6]

What would be displayed in python3 -i for each of the following Python3 expressions?

len(a) 4

a[2] [3,4,5]

a[2][0] 3

a + a [1, 2, [3, 4, 5], 6, 1, 2, [3, 4, 5], 6]

(c) [10 pts] Suppose that we type the following assignments in python3 -i in the given order.

```
x = 5
y = 10
x = x + y
y = y + x
```

For each of the expressions below write down the value that would be outputted if the expression was evaluated in python3 -i after making the assignments above.

x 15

y 25

(y % x) // x 0

4. This question focuses on looping.

(a) [8 pts] We wish to define a Python3 function `out_of_order` that takes an "almost sorted" list as input and returns the first item that is not in ascending order. The function should return None if the list is entirely in ascending order. For example, `out_of_order`([1, 5, 17, 12, 24]) should return 12, since 12 is less than the preceding item, 17. Complete the following iterative function `out_of_order`.

```
def out_of_order(list):
  index = 0
  while index < (len(list)-1) :
    if list[index] > list[index+1] :
      return list[index+1]
    index = index + 1
```

```
  return None
```

(b) [8 pts] Consider the following recursive algorithm for returning the first item in a list that is not in ascending order, else None. Complete the recursive definition of `out_of_order`.

1. If the list has fewer than two elements, return None.
2. If the first element in the list is greater than the second element, return the second element.
3. Otherwise return the result of a recursive call on the tail of the list (i.e., everything beyond the first element.)

```
def out_of_order(list):
  if len(list) < 2 :
    return None
  elif list[0] > list[1] :
    return list[1]
  else:
    return out_of_order(list[1:len(list)])
```

(c) [2 pts] Give an example of a six element list that would be a worst case input for `out_of_order`.

[1,2,3,4,5,6]

(d) [2 pts] What is the big O worst case complexity of `out_of_order`?

O(n)

5. This question deals with searching and sorting.

(a) [2 pts]

What is the big O complexity of binary search? O(log n)

What is the big O complexity of insertion sort? O(n^2)

(b) [6 pts] Fill in the table below to show how binary search would locate the value “e” in the list `["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k"]`. Use the binary search algorithm taught in the book and covered in lecture. Note: this table may contain extra rows.

Iteration	Low	High	Mid	list[mid]
1	-1	11	5	f
2	-1	5	2	c
3	2	5	3	d
4	3	5	4	e

(c) [6 pts] For each sorting algorithm described below, give its correct name:

- For each input item, find its proper position in the result list and add it at that position.

  Insertion Sort

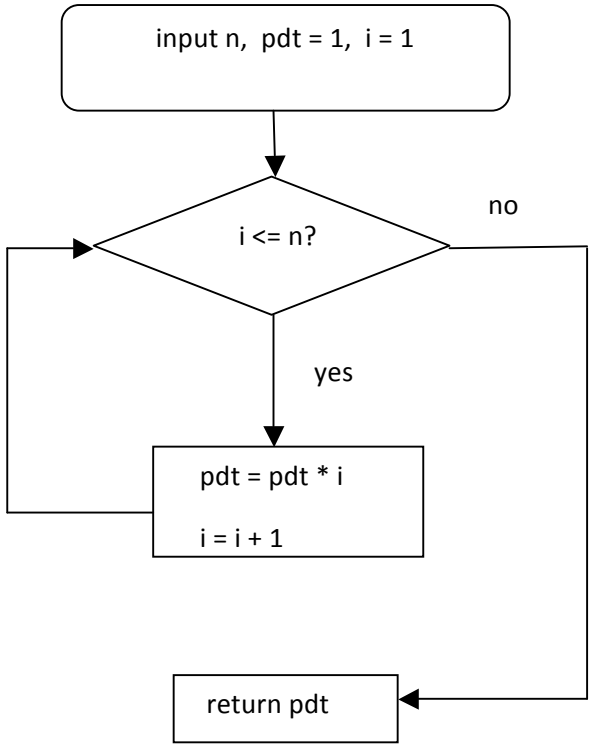
- For each position i in the list, find the index of the smallest item at or to the right of position i, and swap `list[i]` with `list[index_of_smallest]`.

  Selection Sort

- Organize the inputs into N groups of size 1. Systematically combine adjacent groups to form N/2 sorted groups, each of size 2. Repeat the process, combining adjacent groups of size 2 to form N/4 sorted groups of size 4. Keep going until you have one sorted group of size N.

Merge Sort

Consider the following flow chart.



d) [6 pts] Convert the above flow chart into Python3 code. Use the following outline

```
def mystery_function(n):
    pdt = 1
    i = 1
    while (i <= n):
        pdt = pdt * i
        i = i + 1
    return pdt
```

e) [4 pts] Explain in one sentence the purpose of this code (using n in your answer).

The function returns n!.

6. This question is based on your readings from the book *Blown to Bits*.

[6 pts] When you print a report using a laser printer, can you assume that no one can tell who printed it? Give a yes/no answer followed by a one sentence explanation.

No: Each print is associated with a serial number.