Name: _____ Net ID: _____

NYU, Tandon School of Engineering
CS-1134: Data Structures and Algorithms — Fall 2017

CS-1134 – Midterm Exam

Tuesday, October 17, 2017

- You have one hour and 20 minutes.
- There are 7 questions all together, with 100 points total.
- 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 top of each page.
- **Don't use pencils**, as they don't show up well when scanned.
- 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.
- Calculators are not allowed.
- Read every question completely before answering it.
- For any questions about runtime, give an asymptotic analysis.
- You do not have to do error checking. Assume all inputs to your functions are as described
- 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 CS1134 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 (9 points)
For each of the following, state if it is true or false (circle your choice):

I. $\sqrt{3n^2 + 4n - 5} = \theta(n)$ **TRUE / FALSE**

II. $\log_2(n^2) = O(\log_2(n))$ **TRUE / FALSE**

III. $\log_2(n) = \Omega(\sqrt{n})$ **TRUE / FALSE**

Question 2 (12 points)

What is printed when the following Python code is executed?

```
lst1 = [1, [2, 3], [4, 5], 6]
lst2 = lst1
lst3 = lst1[ : ]
lst4 = copy.deepcopy(lst1)

lst1[0] = 10
lst1[1][1] = 30
print("lst1 =", lst1)
print("lst2 =", lst2)
print("lst3 =", lst3)
print("lst4 =", lst4)
```

Output:

_____

_____

_____

_____

_____

Name: _____ Net ID: _____

Question 3 (12 points)

Let *lst* be a list of integers. We define ***prefix_sum(i)*** to be the sum of the first *(i+1)* elements of *lst*. That is: *prefix_sum(i)* = *lst[0] + lst[1] + … + lst[i]*.
For example, if *lst = [-1, 1, 4, -2, -3]*,

- *prefix_sum(0)* is -1
- *prefix_sum(1)* is 0, since (-1) + 1 = 0
- *prefix_sum(2)* is 4, since (-1) + 1 + 4 = 4
- *prefix_sum(3)* is 2, since (-1) + 1 + 4 + (-2) = 2
- *prefix_sum(4)* is -1, since (-1) + 1 + 4 + (-2) + (-3) = -1

Complete the definition below for the function:

```
def positive_prefix_sum(lst)
```

This function is given a list of integers, `lst`. When called, it creates and returns a list with all the indices of which their prefix-sum is positive.

For example, if `lst = [-1, 1, 4, -2, -3]`, calling `positive_prefix_sum(lst)` should return `[2, 3]`.

Notes:

1. Your implementation should all be in the `return` line. That is, you are not allowed to add lines to this function, define an additional function, etc.
2. **You may want to use the build-in `sum` function**.
3. The indices should come in an ascending order.
4. In this question, don't worry about the runtime of your implementation.

```
def positive_prefix_sum(lst):

    return _____
```

Name: _____ Net ID: _____

Question 4 (12 points)

Consider the following two implementations of a function that if given a list, `lst`, create and return a new list containing the elements of `lst` in reverse order.

```
def reverse1(lst):
    rev_lst = []
    i = 0
    while(i < len(lst)):
        rev_lst.insert(0, lst[i])
        i += 1
    return rev_lst

def reverse2(lst):
    rev_lst = []
    i = len(lst) - 1
    while (i >= 0):
        rev_lst.append(lst[i])
        i -= 1
    return rev_lst
```

1. If `lst` is a list of n integers, what is the worst case running time of `reverse1(lst)`? Give a short explanation of your answer.

Your Answer: $\theta($ _____ $)$

_____

_____

_____

_____

2. If `lst` is a list of n integers, what is the worst case running time of `reverse2(lst)`? Give a short explanation of your answer.

Your Answer: $\theta($ _____ $)$

_____

_____

_____

_____

Name: _____ Net ID: _____

Question 5 (10 points)

Below, you are given a recursive implementation for the function:

```
def min_in_lst(lst, low, high)
```

This function gets the list of integers `lst`, as well as two indices: `low` and `high` (`low ≤ high`), which indicate the range of indices of the elements that should to be considered.

When called, the function would return the **minimum value** out of all of elements in the `low, low+1, …, high` positions.

```
def min_in_lst(lst, low, high):
    if(low == high):
        return lst[low]
    else:
        mid_ind = (low + high) // 2
        min1 = min_in_lst(lst, low, mid_ind)
        min2 = min_in_lst(lst, mid_ind + 1, high)
        if(min1 < min2):
            return min1
        else:
            return min2
```

Assuming `lst` is a list of n elements, analyze the worst case running time of the implementation above:

1) Draw the recursion tree that traces the recursive calls of the function.
2) Conclude the total (asymptotic) running time of the function.

Note: Write your answers in the next page.

Name: _____ Net ID: _____

.

i. Draw the **recursion tree** for `min_in_lst(lst, 0, n-1)`

ii. **Conclude the running time** of `min_in_lst(lst, 0, n-1)`

Your Answer: $\theta($ _____ $)$

Calculations:

_____

_____

_____

_____

Name: _____ Net ID: _____

Question 6 (30 points)
Implement the function:

```
def remove_all_evens(lst)
```

This function gets a list of positive integers, `lst`. When called, it should remove all the even numbers from `lst`, and keep only the odd ones.
Note: The relative order of the odd numbers that are left in `lst` at the end, doesn't matter.

For example, if `lst = [2, 3, 5, 2, 16, 13]`, after calling `remove_all_evens(lst)`, `lst` could be the following 3-element list: `[13, 5, 3]`.

Implementation requirements:

1. Your implementation should be in-place.

2. At the end, your list will contain only the odd numbers.

3. Your function should run in **worst case linear time**. That is, if there are n items in `lst`, calling `remove_all_evens(lst)` will run in $\theta(n)$.

4. For the memory used, in addition to `lst`, you are allowed to use only $\theta(1)$ memory. That is, for example, you could **not** use an additional non-constant sized list. But, you could have a few variables.

Note: Write your implementation on the next page.

Name: _____ Net ID: _____

```
def remove_all_evens(lst):
```

Name: _____ Net ID: _____

Question 7 (15 points)

Give a **recursive** implementation for the function:

```
def is_sorted(lst, low, high)
```

This function is given a list of numbers, `lst`, as well as two indices: `low` and `high` (`low ≤ high`), which indicate the range of the indices for the elements that should to be considered.
When called, the function should determine if the elements that are placed at the `low, low+1, …, high` positions, are in an (ascending) sorted order. That is, it should return **`True`** if-and-only-if `lst[low] ≤ lst[low+1] ≤ … ≤ lst[high]`

For example, if `lst = [1, 3, 6, 8, 12, 15, 31]`,
the call `is_sorted(lst, 0, 6)`, will return **`True`**.

Implementation requirements:
Your function should run in **worst case linear time**. That is, if n is the size of the range `low, low+1, …, high`, calling `is_sorted(lst, low, high)` will run in $\theta(n)$.

Note: Write your implementation on the next page.

Name: _____ Net ID: _____

```
def is_sorted(lst, low, high):
```

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.