Exam 1 practice problems

Remember to also pactice: Homework, quizzes, class examples, slides, reading materials.

P1 (MC) For all the questions below (except for the True or False questions), the **answer can be none, one, some or all of the choices**. Write your answers on the **LEFT** side. **No justification needed.** (3 points each question)

a) If *f(N) = O(g(N))*, then *f(N) = Θ(g(N))*. True or False.

b) Insertion sort is $O(N^2)$. True or False.

c) Which of the following is **always** a correct description of the time complexity of the code below (**regardless** of what `someFunction` does)?

A. $\Theta(N)$ B. $O(N)$ C. $\Omega(N)$ D. $O(N\lg N)$

```
int k;
for(k=1; k <= N; k++) {
   someFunction(k);
}
```

d) You are given the option to choose one of three algorithms with time complexities:

A. $\Theta(N^2)$ B. $O(N^2)$ C. $\Omega(N^2)$

You want to choose the algorithm most likely to be the fastest. Which one will you choose?

e) Let $T(N)=\sum_{k=0}^{N}\left(\frac{5}{7}\right)^k=\left(\frac{5}{7}\right)^0+\left(\frac{5}{7}\right)^1+\left(\frac{5}{7}\right)^2+\ldots+\left(\frac{5}{7}\right)^N$. To which of the sets below does *T(N)* belong?

A. $\Theta(1)$ B. $\Theta(N)$ C. $\Theta(N^2)$ D. $\Theta(N\lg N)$ E. $\Theta(\lg N)$

P2. What can you tell about the time complexity of the code below (**regardless** of what `someFunction` does)? Give a lower, upper or tight bound (using **Ω, O,** or **Θ**). **Justify your answer.**

```
int k;
for(k=1; k <= N; k++) {
    someFunction(N);
}
```

P3. Give the **Θ** time complexity for the code below. Justify your answer (clearly show the summation you get and how you solve it, if that is the case).

```
int j, k, t;
for(j = 1; j <= N; j=j+1) {
    for(k = 1; k <= j; k = k+1){
        for(t = 1; t <= N; t=t+1){
            printf("A");
}}}
```

Show step i in the table as well. Add lines to mark different rows:

I	Values of k	Repetitions for k loop	Values of t	Repetitions for t loop (for one k)	Repetitions of printf from loops over k and t for one value of i
step i					

Summation: ……………………………………………………………………….

Closed form solution: ……………………………………………………………… **Θ** (………….)

P4. Suppose that f(N) > 0 for all N >= 0. Suppose that g(N) = f(N)/2 + $\sqrt{N}$. For each of the following, specify if it is "**definitely true**", "**definitely false**", or "**possibly true and possibly false**". **Justify** your answer (using limits or other properties). If you answer "possibly true and possibly false", provide at least one specific example of f(N) that makes the answer "true" and one specific example of f(N) that makes the answer "false".

a) f(N) = O(g(N))

b) f(N) = Θ(g(N))

c) f(N) = Ω(g(N))

P5. Let A = [9, 5, 1, 3, 2, 7]. After 3 complete passes of **insertion sort** (3 iterations of the outer loop) the partially sorted array A is now: [1, 2, 3, 9, 5, 7] .

Is the above statement true or false? Justify your answer.

// Insertion sort pseudo-code provided for reference.

```
for (j = 1; j<= N-1; j++)   // indexes start from 0
    key = A[j]
    i = j-1
    while (i>=0) and (A[i]>key)
         A[i+1] = A[i]
         i = i-1
    A[i+1] = key
```

P6. a) Write code or pseudo-code for insertion sort.

b) Given the array below, show how it is being processed by insertion sort (as described by you above). In particular, on each row, fill in the values of the array at the end of the outer loop of insertion sort.

	4	**9**	**1**	**6**	**11**	**0**	**7**
After the **first** execution of the outer loop.							
After the **second** execution of the outer loop.							
After the **third** execution of the outer loop.							

…(and so on) …							

c) Show the array **after each iteration of the inner** loop.

P7. Show the array, A, at the end of each iteration of the outer loop of **selection sort**.

Index:	0	1	2	3	4	5	6
Original Array:	2	1	3	8	4	6	0

P8. Give a piece of code with nested loops that has time complexity NlgN.

P9. $5N^3 + N^2 = O(N^3)$ True or False? Justify your answer.

P10. $5N^3 + N^2 = \Theta(N^3)$ True or False? Justify your answer.

P11. Given summation: $1 + 2^6 + 3^6 + \ldots + N^6$ Can you solve this in terms of Θ, Ω or O ?

P12. Below is the code for selection sort. There was a suggestion in class for making it stable. Based on that suggestion or your own ideas, modify this algorithm to become stable. (The modified algorithm can behave somewhat different than selection sort, but it should still place the j-th smallest element its final position, j, after the j-th iteration of the outer loop.)

```
int i, j, temp;
for (j = 0; j < N-1; j++)   {
   int min_idx = j;
   for (i = j+1; i < N; i++){
      if (A[i] < A[min_idx])
         min_idx = i;
   }
   temp = A[min_idx];
   A[min_idx] = A[j];
   A[j] = temp;
}
```

b) Give an example of A and show what the original selection sort would do (and be unstable) and what your algorithm does (that it stable).

P13. You have a **sorted** array, A, and an **unsorted** array, B. You need the data from A and B sorted together. Array A has N elements and array B has M elements. You do the following:

- Copy A in a third array, C.
- Insert (in the correct place) every element of B in C. You can assume C is large enough to hold both A and B.

a) (6 points) Give the Θ time complexity for this process. Justify your answer.

b) (6 points) Give an example that shows the best case (use N = 6 and M =4). Give the **values in A and B** (before the processing starts). Give the **Θ time complexity** for your example (which may be different than your answer in part a).

P14. You have to sort an array of bank transactions by date. Most of them are in order (by date), only a few are out of order.

Which sorting algorithm will you use between **insertion sort, selection sort ~~and merge sort~~** in order to take advantage of the fact that the array is almost sorted?

Justify your answer.

Assume that lgN elements are out of order and these lgN elements are placed in the worst possible way for your chosen algorithm. **Both describe the placement** and **give an example** of the placement for an array with N = 8 integers.

What is the time complexity (as a function of N) for your algorithm for the above worst case (of lgN out-of-order elements placed in the worst way in an array of N)?

P15. Letters are pushed on a stack in order: RANDOMOPS. Specify the sequence of pops (shown by a '*') needed to produce the output: ADONOMSPR.

P16. Is this expression fully parenthesized (7-2)+(5+((3*10)-12))?

P17. a) Turn this expression in postfix order ((7-2)+(5+((3*10)-12))). Show the stack at all steps (as done in class and slides).

b) Evaluate this expression 43*92+-593/++. Show stack at all steps as shown in class (see Stack Applications slides).

FOR ALL THE PROGRAMMIMNG PROBLEMS YOU MUST BE ABLE TO ALSO DRAW THE LISTS AND THE ACTIONS AS SHOWN IN CLASS.

P18. Write a function that takes as argument a node `p`, and swaps the two nodes following `p` <u>if the data in the first one is larger than that of the second one</u>. **You must readjust the links, not copy the data from one node into the other**. **No credit given otherwise.**

For example if p -> <u>**6**</u> -> <u>**3**</u> -> 2 -> … the two nodes following `p` have data 6 and 3 and since 6 >3 the nodes will be swapped (keep in mind that you must readjust the links, not just swap the values 3 and 6). If `p` -> <u>**3**</u> -> <u>**6**</u> -> 2 -> … the function will not swap: `p` -> <u>**3**</u> -> <u>**6**</u> -> 2 -> …

Assume the class provided representation of nodes and links:

```
typedef struct node * link;

struct node  {
   int item;
   link next;
};
```

a) The function does not crash (for pointer errors or otherwise). These points are only given if the program is also correct.

b) Draw a picture of what happens with the links when you swap the nodes. Use **line numbers (or code segments) to indicate on the picture** what line of your code produces those changes.

c) Write the function (Do not use anything that would bypass working with the links.)

P19. Write a function **`int triples(int* A, int N)`** that takes as argument an array, `A`, with `N` integers, and returns 1 if all the numbers in A appear a multiple of three times. (That is the same number could appear 3 times or 6 times or 21 times, etc.) Otherwise it will return 0.

You can assume that all the numbers in A are positive (greater or equal to 0).

E.g.: both `triples([5,3,5,3,3,5], 6)` and `triples([5,3,5,3,3,5,5,5,5], 9)` return 1. But `triples([3,7,3,3,7], 5)` returns 0 (7 appears only twice).

a) Give **both** the **time and space complexity** of your program. Justify it by referring back to specific program lines or putting comments in the program.

b) Give a brief but **clear** explanation of how your function works.

c) Write the code. Do all the data manipulation that is needed (if you want to use a specific algorithm, you need to write the code for it).

P20. Write a function that takes **the first nodes** of two lists (A and B) and checks if **each** $B_i = A_1 + A_2 + \ldots + A_i$, where A_1 and B_1 denote the first nodes from the lists. If yes, it returns 1, else it returns 0.
For
A: 3 -> 6 -> 2 -> 5 -> 13 ->1
B: 3 -> 9 -> 11 -> 16 -> 29 -> 30
It returns 1because: $B_1 = A_1 = 3$, $B_2 = A_1 + A_2$ (9 = 3+6),… , $B_6 = A_1 + A_2 + \ldots + A_6$ (30 = 3+6+2+5+13+1)

For
A: **3 -> 6 -> 2** -> 5 -> 13 ->1
B: 3 -> 9 -> **15** -> 16 -> 29 -> 30
It returns 0 because one or more nodes fail the property. In particular, $B_3 \neq A_1 + A_2 + A_3$ (15≠3+9+2),

a) Write the function. (you do **not** need to handle special cases). You should solve it using lists, not by copying the data in arrays and continuing to work with arrays. If you work with arrays, you lose 6 points.
Assume the class provided representation of nodes:

```
typedef struct node * link;

struct node  {
   int item;
   link next;
};
```

b) If your function fails or crashes for certain inputs, give **those inputs** and clearly **indicate the line with the problem** (use line numbers or write the test cases as a comment on that line of code).

c) What is the Θ complexity of your function? Justify your answer.

P21. Graded on correctness, little or no partial credit.

a) Write a function, int check(link first_node), that takes as argument the first **node** of a **single linked list**. It should return ~~1~~ **0** if all the items in the list are unique and ~~0~~ **1** otherwise (if there are repetitions). For example:

For 7-> 4 -> 9 -> 6 -> 4 -> 3   it returns 1 (4 is repeated)

For 5-> 2 -> 9 -> 6 -> 3 it returns 0 (no repetitions).

For a list with only one node it returns ~~1~~ **0** (no repetitions).

Assume that links are implemented using the type and struct given below.

```
typedef struct node * link;

struct node  {
   int item;
   link next;
};
```

b) Give the time complexity of your function in terms of Θ.

P22. Write a function, **int my_count(list L),** that takes as argument a list L of integers (the item is an integer). It should count the number of consecutive repetitions of each item and print both the item and the count. A single occurrence is counted as 0. The function should also return the total number of repetitions. For example, for the list

7-> 2 -> 9 -> 9 -> 9 -> 9 -> 7 -> 7-> 9 -> 3 -> 3   it will print:

```
7, 0
2, 0
9, 3
7, 1
9, 0
```

3, 1
And it will return 5.

Your code should not crash.

Assume that list are implemented using the types and structs defined below. You should do the pointer manipulation (do not assume for example that there is a function that removes a node or one that inserts at a specific position).

```
typedef struct node * link;
typedef struct struct_list * list;


struct node  {
   int item;
   link next;
};

struct struct_list {
   link first;
   int length;
};
```

P23. Assume that you have an implementation of a stack object and you need to simulate a FIFO (First-In-First-Out) queue, using the stack. For simplicity, we will not declare a queue_struct. We will use as the queue a stack object. Provide the implementation of the put_in_FIFO and get_from_FIFO functions (they must have the signature shown here) using the stack's put and get functions.

```
void put_in_FIFO(stack my_queue, int data);
int get_from_FIFO(stack my_queue);
```

From the stack implementation, you only have access to the header file. In particular, you do not know the stack implementation and you do not have access to the stack_struct. You must access the stack **only through the provided functions**:

```
typedef struct stack_struct * stack;


stack newStack(int mx_sz);

void push(stack s, int data);
int ~~push~~pop(stack s);
destroyStack(stack s);
```

```
int isEmpty(stack s);
int getLength(Stack s);
```

The description and the code below are provided to help you understand the problem.

If a stack is accessed through the put_in_FIFO and get_from_FIFO only it should behave like a FIFO queue. For example the client code below would print: **4, 1, 9, 15, 7**

```
int main() {

   stack my_queue = newStack(100);

   put_in_FIFO(my_queue, 4);
   put_in_FIFO(my_queue, 1);
   put_in_FIFO(my_queue, 9);
   int data = get_from_FIFO(my_queue);  // data will be 4
   printf("%d,  ", data);              // prints 4
   put_in_FIFO(my_queue, 15);
   put_in_FIFO(my_queue, 7);

   data = get_from_FIFO(my_queue);
   printf("%d,  ", data);          // prints 1
   data = get_from_FIFO(my_queue);
   printf("%d,  ", data);          // prints 9
   data = get_from_FIFO(my_queue);
   printf("%d,  ", data);          // prints 15
   data = get_from_FIFO(my_queue);
   printf("%d  ", data);           // prints 7
}
```

(5 points) Give the complexity of your functions in terms of θ.