ECE368 Mid-Term Exam 1
Sample Exam

This is a closed book exam. Calculators are not needed nor are they allowed. Since time allotted for this exam is exactly 60 minutes, you should work as quickly and efficiently as possible. *Note the allocation of points and do not spend too much time on any problem.*

Always show as much of your work as practical - partial credit is largely a function of the clarity and quality of the work shown. An answer without justification would not receive full credit. Be concise. It is fine to use the blank page opposite each question for your work.

This exam consists of 11 pages (including this cover sheet and a grading summary sheet at the end). Please check to make sure that all of these pages are present before you begin. Credit will not be awarded for pages that are missing. It is your responsibility to make sure that you have a complete copy of the exam.

IMPORTANT: Write your user (login) ID at the TOP of EACH page. Also, be sure to ***read and sign*** the ***Academic Honesty Statement*** that follows:

"In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing it. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exam and will be subject to possible disciplinary action."

`Printed Name:`

`login:`

`Signature:`

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO ...

1(a) (8 points) Design an algorithm to find both the maximum and minimum of n numbers using at most $\lceil 3n/2 \rceil$ comparisons and no swaps. (To be exact, you can do that using $\lceil 3n/2 \rceil - 2$ comparisons.) You can either describe your algorithm in words or in pseudo-code. For simplicity, assume that n is even.

1(b) (8 points) The following is the pseudo-code for the Selection-Sort algorithm, which uses $(n^2 - n)/2$ comparisons and $n - 1$ swaps to sort an array of n elements.

```
Selection_Sort(A[0..n-1]):
  for i ← n-1 downto 1 {
    max_idx ← 0
    for j ← 1 to i {
      if A[j] ≥ A[max_idx] {
        max_idx ← j
      }
    }
    A[i] ↔ A[max_idx]
  }
```

Modify the Selection-Sort algorithm to use the algorithm you have designed in **1(a)**. Assume that your algorithm in **1(a)** is called `Min_Max(A[lb..ub])`. It finds the indices of the minimum and maximum elements in the subarray `A[lb..ub]` and returns a pair of indices `(min_idx, max_idx)`:

`(min_idx, max_idx)` ← `Min_Max(A[lb..ub])`

1(c) (8 points) For simplicity, assume that the algorithm `Min_Max(`A`[lb..ub])` uses exactly $3(\texttt{ub} - \texttt{lb} + 1)/2$ comparisons to find the indices of the minimum and maximum elements in the subarray `Min_Max(`A`[lb..ub])`. What is the total number of comparisons taken by your Selection-Sort algorithm in **1(b)** to sort $A[0..n-1]$, an array of n elements? Assume that n is even.

2. Stacks and Queues (20 points): Show how to implement a stack using two queues with the following primitives of the queue abstract data type: `EMPTY(Q)`, `ENQUEUE(Q, element)`, and `DEQUEUE(Q)`. In particular, write pseudocode to implement the following stack primitives: `PUSH(S, element)` and `POP(S)`. Assuming O(1) time complexity for all these queue primitives, what is the time complexity of each of the operations `PUSH(S, element)` and `POP(S)`?

3. (18 points) Consider an array-based implementation of singly linked list(s). The contents of the array are as follows:

index	key	next_index
9	A	0
8	B	-1
7	C	3
6	D	8
5	E	7
4	F	-1
3	G	-1
2	H	6
1	I	2
0	J	4

The array contains three lists:

- `Unused_List`: Maintains a list of unused objects. The list starts at index 9.
- `Stack_List`: Implements a stack. The list starts at index 5.
- `Queue_List`: Implements a queue. The list starts at index 1.

3(a) (3 points) Write down the keys (from the head to the tail) in the three lists.

- `Unused_List`:
- `Stack_List`:
- `Queue_List`:

3(b) (3 points) In order to implement all the stack and queue primitives in O(1) time complexity, what other indices, besides the indices pointing to the heads of the three lists, do you have to keep track? The goal here is to minimize the additional space requirement.

3(c) (12 points) Show the contents of the array after performing each of the following three operations in sequence:

1. `Enqueue(Queue_List, Stack_Top(Stack_List))`
2. `Push(Stack_List, K)`
3. `Dequeue(Queue_List)`

Update also the head indices of the lists, as well as the indices you have added in **3(b)**.

	Original		Enqueue		Push		Dequeue	
index	key	next_index	key	next_index	key	next_index	key	next_index
9	A	0						
8	B	-1						
7	C	3						
6	D	8						
5	E	7						
4	F	-1						
3	G	-1						
2	H	6						
1	I	2						
0	J	4						

Head index	Original	Enqueue	Push	Dequeue
Unused_List	9			
Stack_List	5			
Queue_List	1			

Other indices	Original	Enqueue	Push	Dequeue

4. (12 points): Analyze the following function in terms of time complexity:

```
long int Fibonacci(int N)
{
  if ((N == 1) || (N == 0))
    return 1;
  return (Fibonacci(N-1) + Fibonacci(N-2));
}
```

4(a) (6 points): Complete the following table where `Fibonacci()` should represent the number of calls to the Fibonacci function for any given value of N. Also fill in the table with data corresponding to the other functions of N.

N	`Fibonacci()`	N!	2^N	N^2
1	1	1	2	1
2	3	2	4	4
3		6		
4		24		
5		120		
6		720		
7		5040		
8		40320		
9		362880		
10		3628800		

4(b) (2 points): Based on the data of the table above, what is the time complexity of `Fibonacci()` in `O()` notation? Explain your reasoning. Recall that `O()` is a **non-** "tight" upper bound on the complexity of an algorithm.

4(c) (4 points): Briefly explain how you could generate a Fibonacci number in `O(n)` time using only addition.

5. (10 points) A complex number, `(A + jB)`, contains two parts: a real part, A, and an imaginary part, B (where A and B are floating point numbers). Write the specification for a complex number Abstract Data Type (ADT). It should include the value definition and definitions of the following three operators: a constructor for creating the complex number, an operator for addition of two complex numbers, and an operator for multiplication of two complex numbers. Recall that,

```
(A + jB) + (C + jD) = (A+C) + j(B+D)
(A + jB) * (C + jD) = (A*C + B*D) + j(B*C + A*D)
```

6. (16 points) Write pseudo-code using the primitive operations of a singly-linked list to implement `Remove()`, a function that removes the specified node in the list (not the node after). The function must correctly update all links (including links to the first and last nodes) even if there is only one node in the list at the time of the removal. What is the time complexity of the algorithm?

For grading purposes only. Do not write on this page.

Question 1:	/24
Question 2:	/20
Question 3:	/18
Question 4:	/12
Question 5:	/10
Question 6:	/16
TOTAL:	/100