ECE368: Midterm I **Fall 2012**

PUT YOUR LOGIN NAME ON EVERY PAGE OF THE EXAM

Login:_____SOLUTION_____

Name:_____

Read these instructions before going on:

This is a "closed-book," and "closed-notes" examination. No reference materials other than those that are supplied are allowed.

You need only a pencil and an eraser for this examination. If you use ink, use either black or blue ink.

Neatness counts. We will not grade what we cannot read.

Exam is worth 100 points. Some questions may ask for a shorter answer. Please phrase your answer succinctly. Some questions require several intermediate steps. Please show your work. This intermediate work must support the end result in order to obtain any partial credit. Write your answer on the same page, on which the question is given. There are three questions and a total of eight pages. Make sure you turn in all these pages.

This examination contains an amount of material that a well-prepared student should be able to complete in 50 minutes. The exam is to be turned in promptly at the end of the assigned time.

You must demonstrate minimal competence on particular questions (related to **ABET outcomes**) in this examination to pass the class. Those questions are clearly marked.

Read each questions carefully and do only what is specifically asked for in that problem.

Do not attempt to look at other students' work. Keep your answers to yourself.
Any sort of cheating will result in a zero grade.

Read and sign the statement below. Wait for instructions to start the examination before continuing to the next page.

"I signify that the work shown in this examination booklet is my own and that I have not received any assistance from other students nor given any assistance to other students."

_____(Signature)

For grading purposes only...

Problem 1: _____ / 35 (ABET outcome)

Problem 2: _____ / 35 (ABET outcome)

Problem 3: _____ / 30

Total: _____ / 100

ECE368: Midterm I Login Name:_____

Q 1. (35 points) Satisfactory completion of this question satisfies an ABET curriculum outcome for this class (outcome 1). You must earn a score of 60% or higher on this question in order to receive a passing grade for this class.

a. (15 points) Arrange the following functions by order of complexity from low to high.

1. $n \log_2 n$
2. 3^n
3. n^4
4 n
5. **5487**
6. $n\sqrt{n}$

Answer: **5487** n $n \log_2 n$ $n\sqrt{n}$ n^4 3^n

b. (20 points) Given $f(n) = 8n^4 + 2000n$, prove $f(n)$ is $\Theta(n^4)$

Solution: According to the definition, we need to find C_1 , C_2 and n_0, such that:

$0 \leq C_1 n^4 \leq 8n^4 + 2000n \leq C_2 n^4$

$0 \leq C_1 \leq 8 + (2000/n) \leq C_2$

One choice: $n_0 = 2000$, $C_1 = 8$, $C_2 = 9$

Q2 (35 points). Satisfactory completion of this question satisfies an ABET curriculum outcome for this class (outcome 2). You must earn a score of 60% or higher on this question in order to receive a passing grade for this class.

Consider the following recursion for printing a sequence of integers:

$T_1 = 1$

$T_2 = 1 \quad 2 \quad 1$

$T_3 = 1 \quad 1 \quad 2 \quad 1 \quad 3 \quad 1 \quad 2 \quad 1 \quad 1$

Note: in general, for a given n, the sequence can be represented as:

$T_n = \quad T_{n-2} \quad T_{n-1} \quad n \quad T_{n-1} \quad T_{n-2}$

(a) (10 pts) Write a recursive C function named *Seq* that takes *n* as an argument and prints the corresponding sequence. **Use a proper assertion on the argument for *Seq*.**

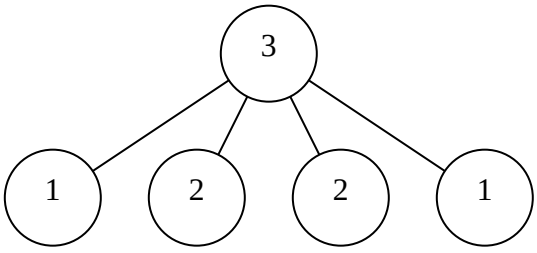
(b) (10 pts) Draw a recursion tree for your function *Seq* generating the sequence when n=3.
(c) (10 pts) How many recursive calls are made in Seq to generate sequence a sequence for n=5?
(d) (5 pts) What is the maximum size of the stack needed to maintain pointers when the recursive function Seq is called by a main program when n =5?

Solution:

(a)
```
void Seq(int n)
{
  assert(n > 0);

  if (n == 1)
    printf("1 ");
  else if (n == 2)
    printf("1 2 1 ");
  else
  {
    Seq(n-2);
    Seq(n-1);
    printf("%d ",n);
    Seq(n-1);
    Seq(n-2);
  }
}
```

(b)



(c) 2*[Seq(4) + Seq(3)] = 2*[(1 + 2*Seq(3) +2*Seq(2)) + Seq(3)] = 2*[1 + 3*Seq(3) +2*Seq(2)]
= 2*[1 + 3*(1+2*Seq(2) +2*Seq(1)) +2*Seq(2)] = 2*[4 + 8*Seq(2) + 6*Seq(1)] = 36

(d) 4

Q3 (30 points). Assume that we have an array of records with DATA, LEFT, and RIGHT fields. Assume that free is the index to the first node in a free list and that head and tail are the indices of a doubly-linked **dequeue (dequeue** is described in Homework 3, as a queue in which items can be added or deleted from either the first or the last position of the queue). The free list is maintained as a singly-linked stack (LEFT field is set to 0, RIGHT field is the index of the next node, and the last node has a 0 in the right field). The LEFT field of the head node of the doubly-linked **dequeue** contains the index of the tail node in the list and the RIGHT field of the tail node contains the index of the head. When the doubly linked **dequeue** is empty, the indices head and tail are set to 0.

The routine **InsertTail** adds a node to the tail of the doubly-linked **dequeue** getting the node from the front of the free list. **InsertHead** adds a node to the head of the doubly-linked **dequeue** getting the node from the front of the free list. The routine **RemoveHead** removes a node from the head of the doubly-linked **dequeue**, returning it to the front of the free list. Assume that when a node is returned to the free list, the DATA and LEFT fields are set to 0. **RemoveTail** removes a node from the tail of the doubly-linked **dequeue**, returning it to the front of the free list.

The initial configuration of a block of memory (representing doubly-linked dequeue) is given below (assuming the **dequeue** is empty):

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	0	0	0	0	0	0	0	0	0	0
LEFT	0	0	0	0	0	0	0	0	0	0
RIGHT	2	3	4	5	6	7	8	9	10	0

```
free = 1            (* the next free location *)
head = tail = 0     (* the queue is empty *)
```

Starting from the above memory configuration, show the memory after each of the following eight commands, executed sequentially **(You can show only those fields that are changed). Also, show the values of free, head and tail after each command.**

Solution:

1) InsertTail 3

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	0	0	0	0	0	0	0	0	0
LEFT	1	0	0	0	0	0	0	0	0	0
RIGHT	1	3	4	5	6	7	8	9	10	0

free = 2 head = 1 tail = 1

2) InsertHead 11

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	11	0	0	0	0	0	0	0	0
LEFT	2	1	0	0	0	0	0	0	0	0
RIGHT	2	1	4	5	6	7	8	9	10	0

free = 3 head = 2 tail = 1

3) RemoveHead

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	0	0	0	0	0	0	0	0	0
LEFT	1	0	0	0	0	0	0	0	0	0
RIGHT	1	3	4	5	6	7	8	9	10	0

free = 2 head = 1 tail = 1

4) InsertHead 5

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	5	0	0	0	0	0	0	0	0
LEFT	2	1	0	0	0	0	0	0	0	0
RIGHT	2	1	4	5	6	7	8	9	10	0

free = 3 head = 2 tail = 1

5) InsertTail 7

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	5	7	0	0	0	0	0	0	0
LEFT	2	3	1	0	0	0	0	0	0	0
RIGHT	3	1	2	5	6	7	8	9	10	0

free = 4 head = 2 tail = 3

6) InsertHead 17

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	5	7	17	0	0	0	0	0	0
LEFT	2	4	1	3	0	0	0	0	0	0
RIGHT	3	1	4	2	6	7	8	9	10	0

free = 5 head = 4 tail = 3

7) RemoveTail

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	5	0	17	0	0	0	0	0	0
LEFT	2	4	0	1	0	0	0	0	0	0
RIGHT	4	1	5	2	6	7	8	9	10	0

free = 3 head = 4 tail = 1

8) RemoveHead

INDEX	1	2	3	4	5	6	7	8	9	10
DATA	3	5	0	0	0	0	0	0	0	0
LEFT	2	1	0	0	0	0	0	0	0	0
RIGHT	2	1	5	3	6	7	8	9	10	0

free = 4 head = 2 tail = 1