King Mongkuts University of Technology Thonburi
Computer Engineering Department
CPE 113 Algorithms and Data Structures
(Sections C and D)
Midterm Examination

26 February 2016 **9:00-12:00**

Student Name_____Student ID_____ Seat _____

Instructions

This examination consists of 6 pages, plus this cover page. ***Do all your work in these examination sheets, in the space provided***.

There are 20 questions. Each question is worth 5 points. Do not spend too much time on any one question. If you get stuck on some question, skip it and come back to it later after you have finished the others.

Read the instructions for each question, and any sample code, very carefully. Some questions ask you to provide reasons for your answers ("why?" or "how do you know?"). If you do not provide a reason, you will not get credit.

Write your answers clearly in the space provided, in English, using pencil or a blue or black pen. For multiple choice questions, ***circle*** the letter of the correct answer(s). Pay attention to the instructions. Some multiple choice questions allow you to choose only one option; some let you choose more than one.

You will have three hours for this examination.

You may use a hard copy or electronic English-Thai or English-French dictionary for this examination. *However, any audio features of your electronic dictionary must be turned off.* No other books or notes are permitted in the examination area. No other electronic devices are permitted.

Instructor

Sally E. Goldin

(Dr. Sally E. Goldin)

1. Assume we are writing code to create a linked list. Here is the definition of the list item structure:

```
typedef struct _item
{
   char itemdata[256];       /* each item holds a string */
   struct _item * next;      /* pointer to the next item in the list */
} ITEM_T;
```

The following function adds an item to the start of a linked list.

```
int insertAtFront(ITEM_T* head, ITEM_T* tail, char* data)
{
   int bOk = 1;  /* we'll return 1 for success, 0 if error */
   ITEM_T * new = (ITEM_T*) calloc(1,sizeof(ITEM_T));
   if (new == NULL)  /* memory allocation failed */
      {
      bOk = 0;
      }
   else
      {
      /* copy string data to the new list item */
      strncpy(new->itemdata,data,sizeof(new->itemdata)-1);
      if (head == NULL)
          head = tail = new;
      else
          {
          /***  MISSING CODE HERE ***/
          }
      }
   return bOk;
}
```

In the space below, write the C statement(s) that should be added instead of the comment "/*** MISSING CODE HERE ***/". (5 points)

2. A list can be programmed using an array or a linked list. What is one reason for using a linked list, instead of an array? (5 points)

3. What does **stack underflow** mean? (5 points)

4. Suppose we start with an empty stack. We execute the following

operations:

```
push(22);
push(19);
pop();
push(39);
push(5);
pop();
pop();
```

After these operations, what does the stack contain? Write the numbers on the stack from left to right on the line below. Assume left is the top of the stack and right is the bottom of the stack. (5 points)

_____

5. Suppose I have a queue with the following contents (start of the queue is at the left):

```
Mary  Joe  Fred  Harry  Lisa
```

I execute the following queue operations:

```
enqueue(Jane)
enqueue(Roger)
dequeue()
dequeue()
enqueue(Mark)
```

On the line below, write the contents of the queue after these operations, with the start of the queue on the left. (5 points)

_____

6. What is the difference between a **doubly linked list** and a regular linked list (singly linked list)? (5 points)

7. Below you will find pseudocode for an algorithm that searches for a target number in a list of numbers sorted from the smallest to the largest value. In this pseudocode, *list* is the list of numbers, *count* is the size of the list, and *target* is the number we are searching for. The algorithm returns true if the number is found in the list, otherwise false. An expression like *list[i]* means the i^{th} element of the list.

```
search(list,count,target)
   if (list[count/2] equals target)
      return true
   else if (count == 1)
      return false
   else if (target > list[count/2])
      return search(list starting at 0,count/2, target)
   else
      return search(list starting at count/2+1,count/2,target);
end search
```

Is this algorithm recursive? How do you know? (5 points)

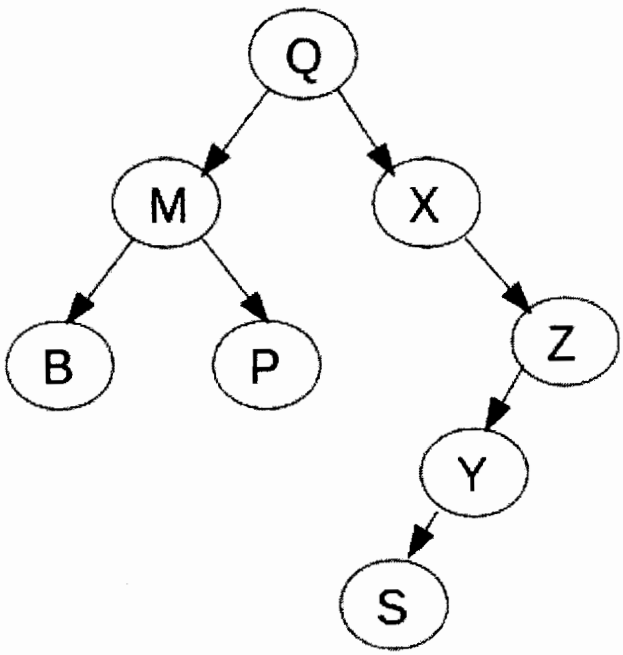
8. What is the big O value (computational complexity) for the algorithm in question 7? (5 points)

9. Suppose we are building a **sorted binary tree**, that is, a tree which follows the rule "smaller to the left, larger to the right". We begin with an empty tree. Then we add values (nodes) in the following order:

   55 78 23 99 3 18 12 44

   In the space below, draw the final tree after inserting these values. (5 points)

10. Suppose we have the following sorted binary tree:



If we do a **Level Order Traversal** of this tree, printing the letter stored at each node as we visit, what will be printed? Write the letters from left to right below, with the first node visited on the left. (5 points)

_____

11. Is the tree in question 10 **AVL Balanced**? Why or why not? (5 points)

12. Consider the tree in question 10. Which of the following is true about node **B** and node **Z**? You can circle multiple letters. (5 points)

    a. B and Z are siblings
    b. B and Z are at the same level of the tree
    c. B and Z are both leaf nodes
    d. B and Z are both descendants of Q
    e. B and Z are both ancestors of Q

13. In the **couples.c** program we did in Lab 2, we had a structure to represent a single person, like this:

```
typedef struct _person
{
   char name[64];      /* Name of person */
   char gender[2];     /* Gender (M or F) */
   struct _person * partner;  /* Pointer to the other person in a couple */
                       /* NULL if the person is not part of a couple */
} PERSON_T;
```

Suppose I want to write a function that will make two people (represented by **PERSON_T** records) into a couple. It might look like this:

```
/* Make personA and personB into a couple.
 */
void makeCouple(PERSON_T* personA, PERSON_T* personB)
{
    /**** MISSING CODE HERE *****/
}
```

In the space below, write the C statements that should replace "/**** MISSING CODE HERE ****/" in order to implement the function. (5 points)

14. Suppose we have a program that has built a tree. When the program has finished, we want to free the memory in the tree. What kind of tree traversal should we execute to do this? (Your answer should be the one of the four tree traversal algorithms we have studied.) (5 points)

15. What is an **Abstract Data Type** (ADT)? (5 points)

16. In the demos from the first lecture, we found that **treeSort** was much more efficient than **dynamicBubble** in sorting the same data. Why is this true? (5 points)

17. Which of the following describes the data structure you used in Lab 6

(**dictionary.c**). Circle only one letter. (5 points)

a. A linked list of dictionary entries, each of which stored information about one word
b. An array of dictionary entries, each of which stored information about one word.
c. A linked list in which each item was a linked list of dictionary entries.
d. An array of linked lists in which each list stored all entries for one letter of the alphabet.
e. A linked list of arrays, in which each array stored all entries for one letter of the alphabet.
f. None of the above.

18. The list below contains different "Big O" values. Rank them from most efficient to least efficient by putting **1** next to the most efficient, **2** next to the second most efficient, and so on. (5 points)

- $O(n)$ _____
- $O(n^2)$ _____
- $O(n log_2 n)$ _____
- $O(log_2 n)$ _____
- $O(2^n)$ _____

19. When we are talking about recursion, what is the meaning of the term **base case**? (5 points)

20. What is the topic of your video? (5 points)