- 3 [illegible]

King Mongkuts University of Technology Thonburi
Computer Engineering Department
CPE 111 Programming with Data Structures
(Sections C and D)
Final Examination

17 May 2017 **9:00-12:00**

Student Name_____Student ID_____ Seat _____

Instructions

This examination consists of 9 pages, plus this cover page. ***Do all your work in these examination sheets, in the space provided***. You may use the back of the exam as scratch paper.

There are 20 questions. Each questions 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 any 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).

You will have three hours for this examination.

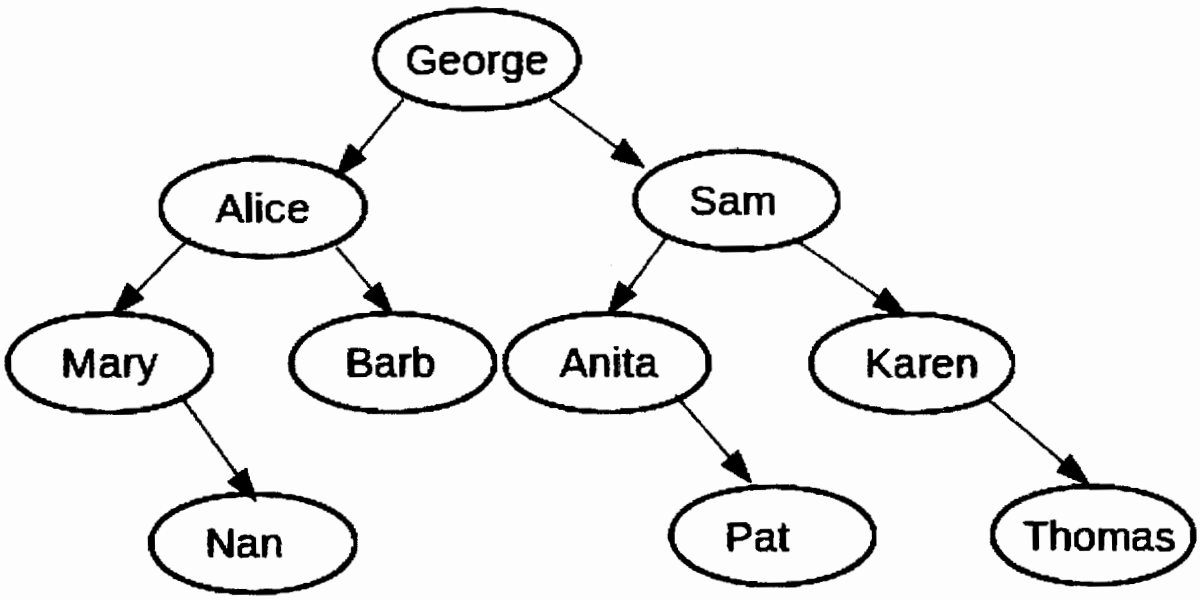
You may use a hard copy or electronic English-Thai or English-French dictionary for this examination. *However, any audio features of your electronic dictionary must be turned off.* No other books or notes are permitted in the examination area. No other electronic devices are permitted.

Instructor

Sally E. Goldin

(Dr. Sally E. Goldin)

1. Study the tree shown in the picture below.



Is this a ***sorted binary tree***? Why or why not? (You will not get any credit if you do not give a correct reason for your answer.) (5 points)

2. Is the tree in Question 1, ***AVL Balanced***? Why or why not? (You will not get any credit if you do not give a correct reason for your answer.) Remember that AVL balanced trees have a very specific definition for "balance". (5 points)

3. Suppose you do a ***post-order traversal*** of the tree in Question 1. In the space below, write the keys for the nodes in the order you would visit them (left to right). (5 points)

4. Suppose we have a ***doubly linked list*** that is composed of list nodes with the following structure:

```
typedef struct _listnode
{
   void * listData;             /* pointer to any sort of data */
   struct _listnode * next;     /* link to next node */
   struct _listnode * prev;     /* link to previous node */
} LISTNODE_T;
```

I also have two global pointers to the head and tail of the list:

```
LISTNODE_T * head;
LISTNODE_T * tail;
```

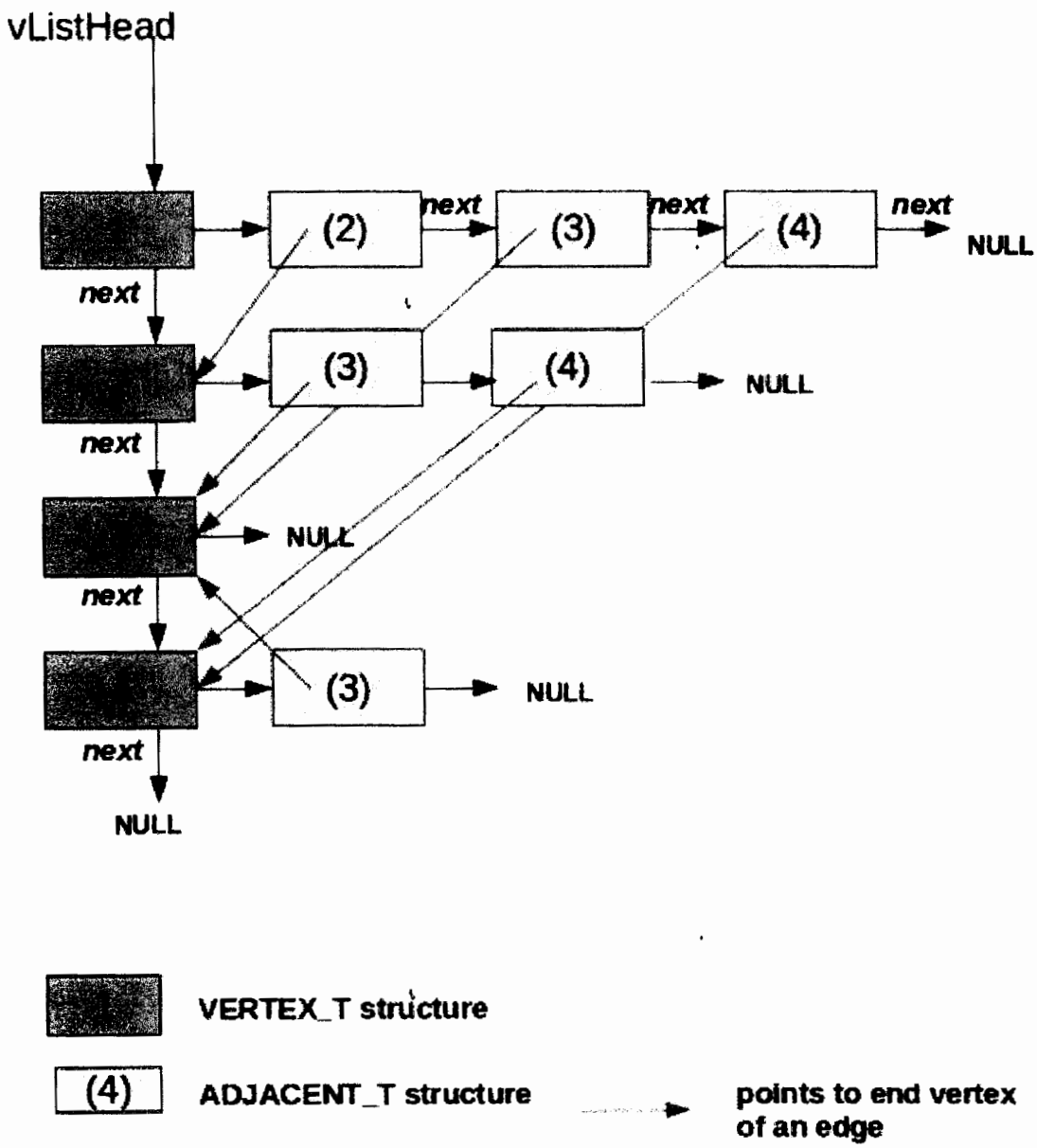
I allocate new list node as follows:

```
LISTNODE_T * newNode = (LISTNODE_T*) calloc(1,sizeof(LISTNODE_T));
```

In the space below, write correct C code to add **newNode** to the end of the list. You can assume there is already at least one item in the list. You only need a few statements. (5 points)

5. The abstract data type List can be implemented in a variety of ways, including as an array or as a linked list. You, as the programmer, need to choose the best implementation for your problem. What is one reason for choosing a ***linked list***? (5 points)

6. The diagram below shows a graph implemented as an adjacency list.



In the space below, **draw this graph**, correctly showing the vertices and the edges (5 points)

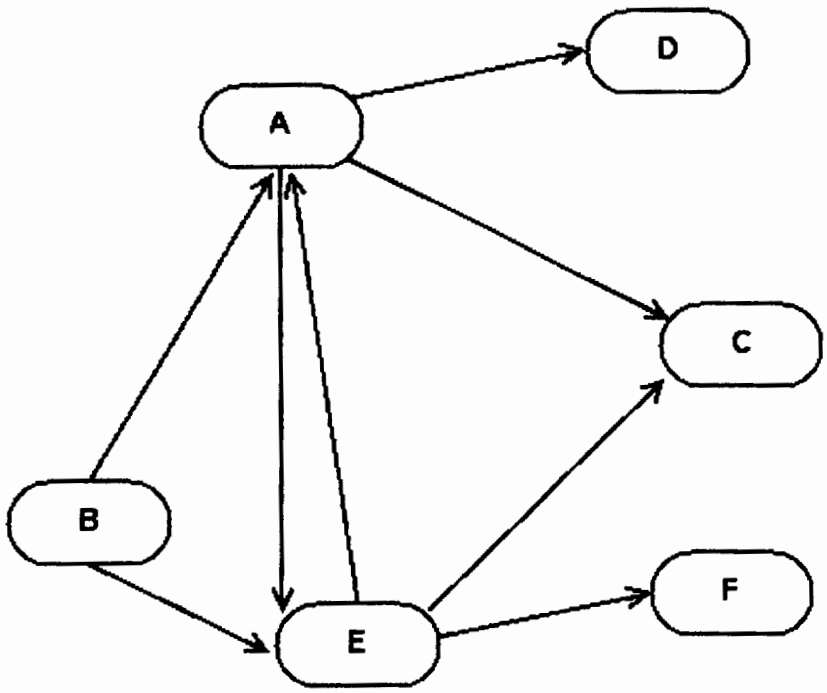
7. Suppose I have a ***min priority queue***. I execute the following operations, in the following order:

```
enqueueMin(30)
enqueueMin(23)
enqueueMin(14)
enqueueMin(49)
dequeueMin()
enqueueMin(27)
dequeueMin()
```

Now I call **dequeueMin()** again. What value will the min priority queue return? (5 points)

8. What is the difference between a graph and a network? (5 points)

9. Suppose I have the following graph:



I want to find the shortest path from Vertex B to Vertex F, where "shortest" means the path that traverses the smallest number of edges. What algorithm should I use? (5 points)

10. Suppose I have a ***max heap*** implemented as a partially sorted tree, as we used in the **emergency.c** lab assignment. (A max heap is the opposite of a min heap; each time you call **heapExtract** you get the largest value currently in the heap.) The contents of the tree, represented as an array, are as follows. (The top row of numbers are the index values into the array.)

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
99	**54**	**32**	**41**	**45**	**28**	**19**	**13**	**27**	**6**	**15**	**18**	**11**	**17**

When we call **heapExtract()**, the heap will return the root, largest value, which is 99. Then the heap will be rearranged so that the next largest value becomes the root. Fill in the table below with the correct arrangement of values, after this rearrangement. (5 points)

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*

11. What is the main **difference** between a graph and a tree? (5 points)

12. In the Java programming language, what is ***ArrayList***? (Hint: we used it in the last lab) (5 points)

13. You were required to read the article "Comments Are More Important Than Code" by Jef Raskin. Which of the following statements summarizes this article best? (5 points)

    a. Modern automatic documentation tools make comments by humans unnecessary.
    b. Code that uses clear formatting and meaningful variable names can reduce the need for comments.
    c. Programmers should create detailed comments explaining not only what their code is doing but why, because no tool can do this.
    d. Making programmers write lengthy comments is a waste of time.
    e. The best documentation is a comment at the end of each line of code, explaining what the code is doing..
    f. None of the above is a good summary.

14. Who is Jef Raskin and why is he famous? (5 points)

15. Suppose I have a hash table that has 7 slots. I am using the following hash function:

```
#define TABLESIZE 7
int myHash(char *string)
    {
    return (strlen(string) % TABLESIZE);
    }
```

    I insert the following words into my hash table:
    - crocodile
    - hippopotamus
    - ant
    - cat
    - tiger
    - buffalo
    - giraffe

    After I insert these words, how many words will be stored in hashtable bucket number 5 (*table[5]*)? (5 points)

16. Suppose I start with an empty **stack**. I execute the following operations:

```
push("Fred")
push("is")
push("hungry")
pop()
push("happy")
pop()
push("a")
push("computer")
push("engineer")
```

After these operations, what is stored on the stack? Write the contents below, assuming the bottom of the stack is on the left and the top of the stack is on the right. (5 points)

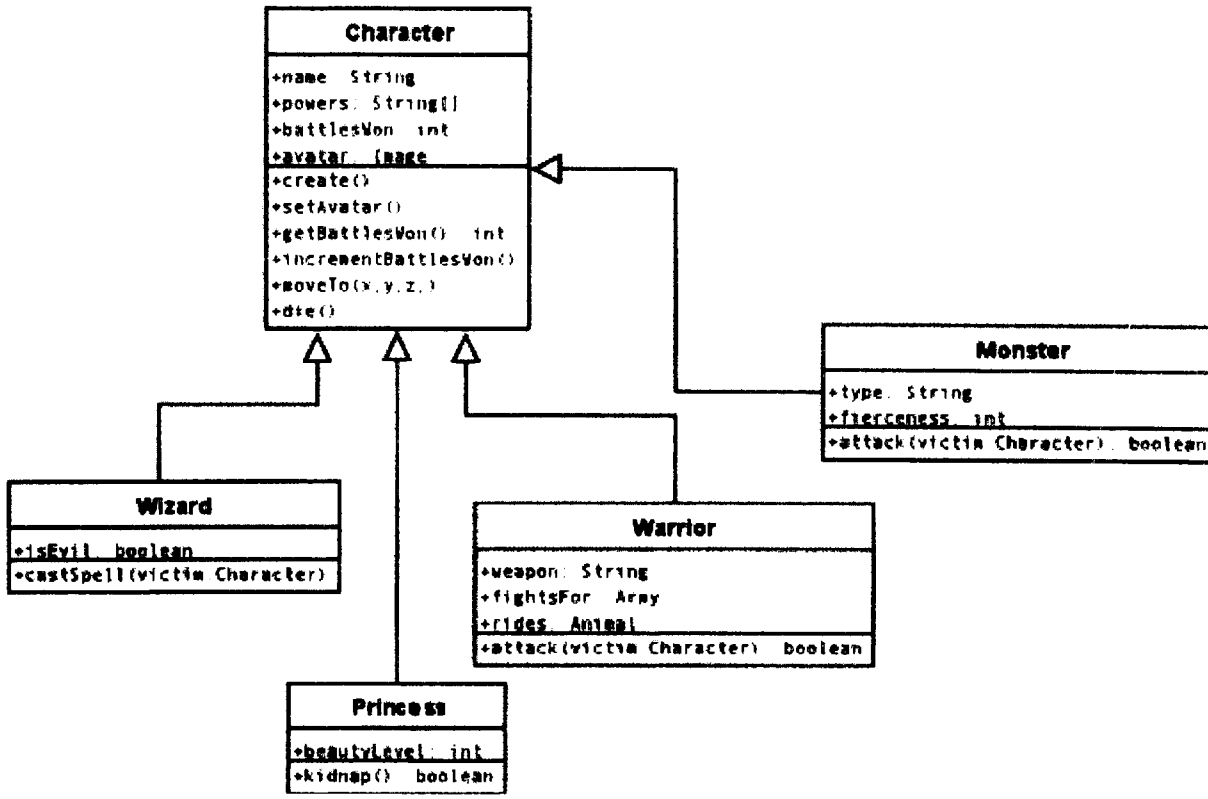
17. Is **Dijkstra's Algorithm** for finding shortest paths in a directed network a recursive algorithm? How do you know? (5 points, no credit unless you explain your answer)

18. Suppose we are building a tree data structure. Our tree nodes look like this:

```
typedef struct _treenode
{
    char keyvalue[256];
    void* otherData;
    struct _treenode ** pChildren;
} TREENODE_T;
```

What is the maximum number of children a node in this tree can have? (5 points)

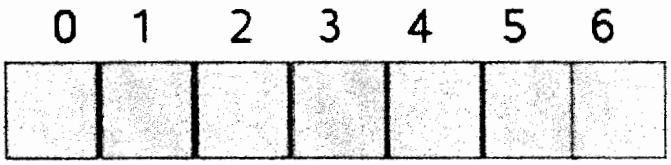
19. Study the following class diagram, which shows part of a class hierarchy in an online role playing game:



Which of the following is a true statement about the classes in this diagram? Circle all statements that are true. (5 points)

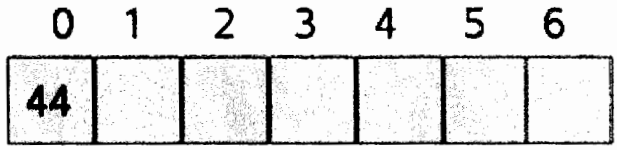
a. *Warrior* is a subclass of *Wizard*
b. A *Princess* can have *powers*
c. *Monster* is a subclass of *Character*
d. We can call the function *isEvil* to find out whether a *Warrior* is good or bad.
e. None of these statements is true.

20. The code on the next page is taken from my demo **arrayQueue.c**. If you remember, this program used a fixed size "circular" array to implement the queue. The pictures on the next page illustrate this.



headIndex = 0
tailIndex = -1
count = 0

enqueue(44)



headIndex = 0
tailIndex = 0
count = 1

To test your understanding of how this code works, I have **removed one important statement from this code**.

```
#define MAXSIZE 10
int headIndex = 0;   /* index into the array where first item in the queue
                      * is located, the one that will returned by dequeue()*/
int tailIndex = -1;  /* index into the array where the last item in the
                      * queue is currently located.*/
int count = 0;       /* number of items currently stored in the queue */
void * queue[MAXSIZE] = {NULL}; /* the queue array */

/**
 * Add a data item to the queue (end of the list)
 * Arguments:
 *   data      -   Pointer to generic data we want to add to queue
 * Returns 1 if successful, 0 if we have run out of space.
 */
int enqueue(void* data)
{
   int bOk = 1;                          /* 1 */
   /* If there is room */                /* 2 */
   if (count < MAXSIZE)                  /* 3 */
      {                                  /* 4 */
      queue[tailIndex] = data;           /* 5 */
      count++;                           /* 6 */
      }                                  /* 7 */
   else                                  /* 8 */
      {                                  /* 9 */
      bOk = 0;                           /* 10 */
      }                                  /* 11 */
   return bOk;                           /* 12 */
}
```

What statement was removed? Where should it be added (after which numbered statement)? (5 points)