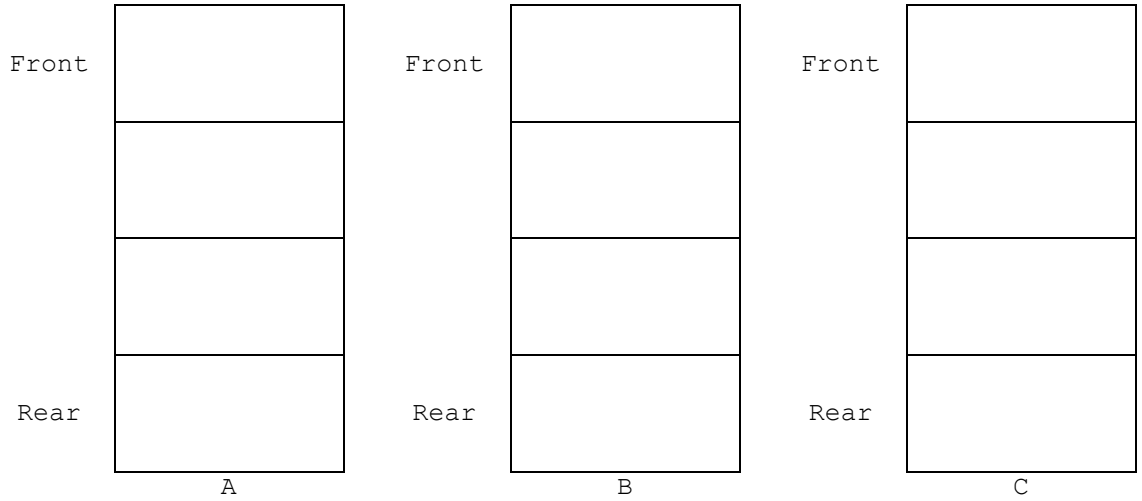
3) (5 pts) ALG (Queues)

Consider the following C code that represents a FIFO queue that holds a list of superheroes as strings. Show the contents of the queue after each indicated point commented (A, B, and C).

```
typedef struct node_s {
    char * hero;
    struct node_s * next;
} node_t;

typedef struct {
    node_t * front;
    node_t * back;
    int size ;
} queue_t;

void enqueue(queue_t * heroqueue, char * hero);
char * dequeue(queue_t * heroqueue);

void followQueue(queue_t * heroqueue){
      enqueue(heroqueue, "Hawkeye");
      enqueue(heroqueue, "Thor");
      enqueue(heroqueue, "Spider-Man");
      dequeue(heroqueue);
      enqueue(heroqueue, "Wanda");
      enqueue(heroqueue, "Vision"); //A
      enqueue(heroqueue, "Ms. Marvel");
      enqueue(heroqueue, "Dr. Strange");
      dequeue(heroqueue);
      dequeue(heroqueue);
      enqueue(heroqueue, "Loki");
      enqueue(heroqueue, "Captain Marvel");
      dequeue(heroqueue);
      dequeue(heroqueue); //B
      enqueue(heroqueue, "Iron Man");
      dequeue(heroqueue); //C
}
```



Note: There are exactly the correct number of boxes to be filled for each state. But, the intermediate steps may have the queue store more than 4 elements.

Computer Science Foundation Exam

August 24, 2024

Section B

ADVANCED DATA STRUCTURES

NO books, notes, or calculators may be used,
and you must work entirely on your own.

PLEASE USE CAPITAL LETTERS IN WRITING YOUR NAME

Last Name: _____

First Name: _____

UCFID: _____

Question #	Max Pts	Category	Score
1	10	DSN	
2	5	ALG	
3	10	ALG	
TOTAL	25	----	

You must do all 3 problems in this section of the exam.

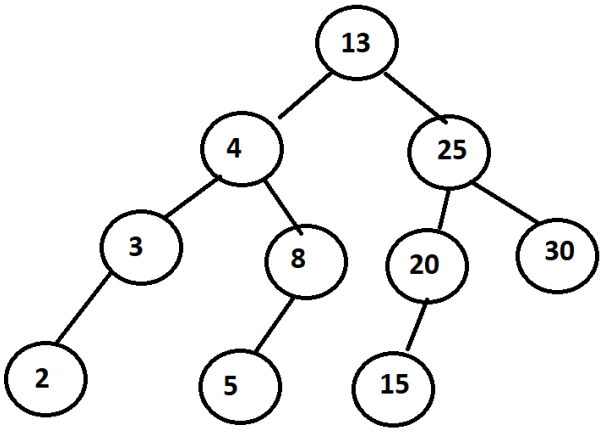
Problems will be graded based on the completeness of the solution steps and <u>not</u> graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all <u>be neat</u>. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

1) **1**(10 pts) DSN (Binary Trees)

Consider the following mystery function that uses a typical tree node structure of a Binary Tree.

```
struct treenode {
    int data;
    struct treenode *left;
    struct treenode *right;
};

int mystery (struct treenode* root) {

    if(root == NULL)
        return 0;

    if(root->data %2 == 0) {
        struct treenode* temp = root->left;
        root->left = root->right;
        root->right = temp;
    }

    return 5 + mystery(root->left) + mystery(root->right);
}
```

a) Redraw (on the right side) the state of the tree below after mystery is called on its root, AND **b)** indicate the value returned by the function.



b) Return Value: _____