COP 3502 Section 1 Exam 1A – Linked Lists, Stacks, Queues (Wednesday 10/11/2023)

Last Name: ___________________, First Name: ______________________

Lab Number: _________________ (Must be 11 – 15 and match entry "LabNum" in Webcourses)

1) (15 pts pts) Imagine using a linked list to store UCFID numbers for students. For this question, we'll use the following struct which defines how a node is stored for this list:

```
typedef struct node {
    char name[50];
    int ucfid;
    struct node* next;
} node;
```

Writing a function called `change`, which takes in a pointer to the front of the linked list storing the data, the name of a student, their UCFID number. If a node in the list stores the student's name (exactly as defined by strcmp), then the corresponding ucfid field in that node should be changed to the formal parameter, `newid`. If no node in the list stores the formal parameter, `newname`, then a new node should be created, with the name and number copied into the node appropriately, then the node should be inserted into the front of the existing list, and the pointer to the new front of the list should be returned.

```
node* change(node* sList, char newname[], int newid) {













}
```

2) (9 pts) Consider implementing a queue with an array. An incomplete implementation is provided below (that may be slightly different than the one shown in class**). Please complete the dequeue function on the next page.** The specification for what you should do in the function is indicated in the header comment for the function. Note: This queue does not support growing indefinitely. The queue will not support the enqueue operation if it has 10 items in it (full).

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

typedef struct queue {
    int items[MAX];
    int front;
    int cursize;
} queue;

int full(queue* q) {
    return q->cursize == MAX;
}

int empty(queue* q) {
    return q->cursize == 0;
}

void init(queue* q) {
    for (int i=0; i<MAX; i++) q->items[i] = 0;
    q->front = 0;
    q->cursize = 0;
}

int enqueue(queue* q, int value) {
    if (full(q)) return 0;
    q->items[(q->front+q->cursize)%MAX] = value;
    q->cursize++;
    return 1;
}
```

```
// If the queue pointed to by q is empty, 0 is returned and no action
// is taken. Otherwise, the value to be dequeued is stored in the
// variable pointed to by retval, and the value is removed from the q
// pointed by q, and 1 is returned to indicate a successful dequeue.
int dequeue(queue* q, int* retval) {








}
```

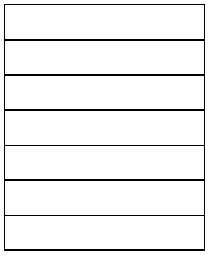
3) (8 Pts) The function below takes in a pointer to a linked list and potentially modifies some of its contents. What would the contents of the list, `mylist`, be right after the function call `mystery(mylist)` executes? Right before the call, here is a picture of the contents of the list pointed to by `mylist`:

```
mylist → 8 → 4 → 12 → 13 → 2 → 9 → 19 → 18

typedef struct node {
    int data;
    struct node* next;
} node;

void mystery(node* list) {
    if (list == NULL || list->next == NULL) return;
    if (list->data > list->next->data) {
        list->next->data += list->data;
        list->data = list->next->data - list->data;
        list->next->data -= list->data;
    }
    mystery(list->next);
}
```

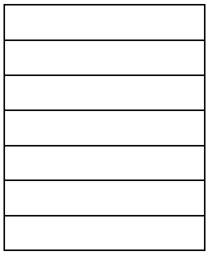
mylist → ____ → ____ → ____ → ____ → ____ → ____ → ____ → ____

4) (7 pts) Evaluate the following postfix expression, showing the state of the stack used in the algorithm at the two given checkpoints, A and B. (I usually ask three, but this time I'm just asking for two checkpoints.)

```
                                                         A                 B
12       3        /        16       9        +        8       3        –        /        3        *        +
```



A



B

Value of Expression = ___________

5) (1 pt) What diary product is in a Grilled Cheese sandwich? ____________________