COP 3502 Section 4 Exam 1A – Linked Lists, Stacks, Queues (Thursday 10/11/2023) Solution

1) (9 pts) Convert the following expression in infix notation to postfix notation using the algorithm shown in class. Please indicate the contents of the stack used in the algorithm at each of the points indicated in the expression (A, B and C):

(5 + 7) / ((2 + [A] 6 * 3 – 4 * (9 / 3) [B]) / (1 + [C] 1))

A	B	C
	*	+
+	–	(
(	(	/
(	(	(
/	/	/

Postfix Expression:

5 7 + 2 6 3 * + 4 9 3 / * – 1 1 + / /

Grading: 2 pts each stack (give 1 pt if mostly right), 3 pts total expression (give partial if necessary 2 pts if only one error, 1 pt if more than half right)

2) (5 pts) In the recursive fast modular exponentiation function, the most number of recursive calls on the call stack when the initial function call passes in the exponent 27 is 8. In the stack below, place each exponent for each distinct recursive call. The bottom number has been placed for you already.

exp = 1
exp = 2
exp = 3
exp = 6
exp = 12
exp = 13
exp = 26
exp = 27

Call Stack

Grading: 5 pts if fully correct, 4 pts if 6 of the 7 correct numbers show up in decreasing order going up the stack, 3 of 5 pts if 4 or 5 corrrect values show up in order, 2 of 5 if 2 or 3 values show up in order, 1 of 5 if at least 1 correct # shows up and list is in order, 0 of 5 if list is not strictly decreasing.

3) (15 pts) In this question you will write a **recursive** function that takes in a pointer to a linked list and reverses all of the links in the list, returning a pointer to the new front of the list, which has the nodes in the reverse order of where they were previous to the function call. The strategy you will use is as follows:

1) Return a pointer to the front of the list if it's size 0 or size 1.
2) Recursively reverse "the rest of the list" (except the first element), storing the pointer returned from this recursive call in a new variable. (The front of this list is also the front of the list you must return.)
3) Iterate through the list returned in step 2 until arriving at a pointer to its last node. (This last node used to be the second node in the original list.)
4) Link, the node pointed to by the pointer described in step three to the first node in the original list.
5) Set the next pointer for the original first node to NULL.
6) Return the new front of the list.

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

node* reverse(node* list) {

    // Grading: 3 pts
    if (list == NULL || list->next == NULL) return list;

    // Grading: 3 pts
    node* rest = reverse(list->next);

    // Grading: 1 pt.
    list->next = NULL;

    // Grading for getting to last item - 5 pts.
    node* tmp = rest;
    while (tmp->next != NULL) tmp = tmp->next;

    // Grading: 2 pts
    tmp->next = list;

    // Grading: 1 pt
    return rest;
}
```

4) (10 pts) Imagine processing a queue of **positive** integers as follows:

1. Start a counter at 1.
2. Repeat the following steps until the queue is empty:
    a. Dequeue the front item in the queue.
    b. If the counter is not a multiple of the number dequeued, enqueue that number to the back of the queue, otherwise print out the dequeued number on a line by itself.
    c. Add 1 to the counter.

You are given the following queue functions:

```
void init(struct queue* qPtr);
int enqueue(struct queue* qPtr, int val);
int dequeue(struct queue* qPtr);
int empty(struct queue* qPtr);
```

Use these to write a function that executes this simulation. The queue will be passed into the function.

```
void simulate(struct queue* qPtr) {

    int cnt = 1;                                    // 1 pt
    while (!empty(qPtr)) {                          // 1 pt

        int cur = dequeue(qPtr);                    // 2 pt
        if (cnt%cur == 0)                           // 2 pts
            printf("%d\n", cur);                    // 1 pt
        else                                        // 1 pt
            enqueue(qPtr, cur);                     // 1 pt

        cnt++;                                      // 1 pt
    }



}
```

5) (1 pt) What color is the White House? **WHITE (Give to all)**