COP 3502 Quiz #2 Version C (Recursion, Linked Lists, Stacks, Queues, SLMP) Solution

1) (5 pts) Consider generating a sequence of positive integers starting with a given value ***n*** as follows: if n is odd, make the next integer 5n+1. If n is even, make the next integer n/2. Continue generating terms in the sequence until the value 1 is generated. For example, if the starting term is n = 12, then the sequence generated is 12 → 6 → 3 → 16 → 8 → 4 → 2 → 1, which has 8 terms. Write a ***recursive*** function that takes in ***n*** and returns the number of terms in the sequence generated in this fashion, which starts with n. You may assume the function gets called with a value of n for which the sequence length is finite and the maximum term of the generated sequence fits in an integer.

```
int numTerms(int n) {

    // Grading: 1 pt
    if (n == 1) return 1;

    // Grading: 2 pts
    if (n%2 == 0) return 1 + numTerms(n/2);

    // Grading: 2 pts
    else return 1 + numTerms(5*n+1);

}
```

2) (5 pts) Evaluate the following postfix expression, showing the state of the stack at each of the indicated points.

						A					B						C					
8	**3**	**5**	**6**	**-**	**-**		**/**	**6**	**3**	**2**		**+**	*****	**3**	**7**	**+**		**/**	**4**	**+**	*****	

A	B	C
	2	
	3	**10**
4	**6**	**30**
8	**2**	**2**

Value of the Expression: **14**

Grading: 1 pt for each stack (all or nothing), 2 pts for the final answer (all or nothing)

3) (8 pts) Consider using a linked list of digits to store an integer. Specifically, the integer 2471 would be stored as the following linked list: 1 → 7 → 4 → 2. (Thus the first node stores the units digit, the second node the ten's digit, the third node the hundred's digit and so forth.) Write a ***recursive*** function that takes in a pointer to an integer stored in this manner as well as an integer location (0 or higher) and returns the corresponding digit in that number located in the $10^{location}$ place. If no such digit exists, -1 should be returned.

```
typedef struct numNode {
    int digit;
    struct node* next;
} numNode;

int getDigit(numNode* num1, int location) {

    // Grading: 2 pts
    if (num1 == NULL) return -1;

    // Grading: 3 pts
    if (location == 0) return num1->digit;

    // Grading: 3 pts
    return getDigit(num1->next, location-1);
}
```

4) (5 pts) What problem does the function below solve? Namely, what is the return value of the function in relation to the input parameters? Your answer should be a regular sentence or two in English and NOT a literal translation of the code. The latter will receive **no credit!**

```
// Pre-condition: array is size n and sorted with unique values.
//                target > 0.
int mysteryC(int* array, int n, int target) {
    int res = 0, low = 0, high = 0;
    while (low < n) {
        if (high == n || array[high] - array[low] > target) {
            res += (high-low);
            low++;
        }
        else
            high++;
    }
    return res;
}
```

This function returns the number of subsections of the array of size 1 or greater where the difference between the largest and smallest element is target or less. A subsection is simply a set of consecutive terms in the array. For example, if the array were [2, 5, 8, 12] and target = 6, then the subsections in the array that would be counted are [2], [2, 5], [2, 5, 8], [5], [5, 8], [8], [8, 12] and [12].

Grading: 2 pts for mentioning subsections or the array (any reasonable wording here is fine as long as you get the feeling they understand the idea), 2 pts for mentioning the difference between terms, 1 pt for putting it all together.

5) (5 pts) The permutation algorithm shown in class generates the permutations of {0, 1, 2, …, , n-1} in lexicographical order. Which permutation does the algorithm generate after the permutation shown below?

4, 2, 8, 0, 5, 9, 7, 6, 3, 1

4, 2, 8, 0, 6, 1, 3, 5, 7, 9

Grading: 2 pts for keeping 4, 2, 8, 0 the same. 1 pt for the placement of 6, 2 pts for the correct arrangement of 1, 3, 5, 7, 9

6) (7 pts) In the implementation of a queue using an array shown in class, the following four components formed the structure storing the queue:

```
elements - an array storing the items in the queue.
front - index of the array that is the front of the queue
numElements - number of elements currently in the queue.
queueSize - currently allocated space for array elements.
```

Assume that the queueSize is initially set to 10 and that the following code segment is run:

```
struct queue* MyQueuePtr = malloc(sizeof(struct queue));
init(MyQueuePtr);

for (int i=0; i<7; i++) {
    enqueue(MyQueuePtr, 4*i+5);
    enqueue(MyQueuePtr, 2*i+4);
    int tmp = dequeue(MyQueuePtr);
}
```

Show the value of each of the four components of the struct pointed to by MyQueuePtr right after this code segment is executed.

front **7** numElements **7** queueSize **10**

index	0	1	2	3	4	5	6	7	8	9
elements	**25**	**14**	**29**	**16**				**10**	**21**	**12**