Data Structure - Exam I

2020/10/15

- Note: **The only acceptable programming language in your answer is C.**

1. (15%) Give the order of complexity O(•) of the following expressions.
   (a) $2^n + n^2$
   (b) $n^{1/2} + n \log n$
   (c) $\sum_{i=0}^{n} x^i$

2. (10%) Give a declaration (i.e., type definition) to the structure of the following data type Employee. (In order words, after this structure "Employee" is defined, we can set for example "Employee.Salary = 30000" and "Employee.Dependent.Spouse.Sage = 50", etc.)

Employee

Name (8 chars)	SSN (integer)	Salary (integer)	Spouse	
			Sname (8 chars)	Sage (integer)

3. (25%) Answer the following subquestions.
   (a) Transfer the infix a*b*c/((d+e*(f-g))-h) to postfix expression and give the detailed steps of how your answers are obtained. (10%)
   (b) Transfer the postfix abc/+def+*gh-/-j+ to infix expression and give the detailed steps of how your answers are obtained. (10%)
   (c) What is the advantage of using a stack to evaluate a postfix expression? Explain the reason. (5%)

4. (10%) Given a string S = *a b c a b a b c a b* and a pattern P = *a b c a b c*, use the KMP algorithm to search whether P can be found in S. Detailed steps have to be given to get any score.

5. (10%) The following is a C code segment. Fill in appropriate instructions in the blank spaces so that the linked list is inverted.

```
struct Node {
    int data;
    struct Node* next;
};

/* Function to invert a linked list */
static void reverse(struct Node** head_ref)
{
    struct Node* prev = NULL;
    struct Node* current = *head_ref;
    struct Node* next = NULL;
    while (current != NULL) {
        // Reverse
            (1)          ;
            (2)          ;
            (3)          ;
            (4)          ;
     }
         (5)          ;
}
```

6. (30%) Answer the following subquestions about a circular queue.

(a) (10%) Define the data type of a circular queue. (Not just draw a diagram. You need to define the data type in order to get any score.)

(b) (20%) Fill in the blank spaces in the following program segments so that adding/deleting an item to/from a queue can be correctly functioned.

```
//MAX-QUEUE-SIZE is the size of the queue
element queue[MAX-QUEUE-SIZE];

/* Adding an item to a queue */
void addq(int front, int *rear, element item)
{
```

```
    [        (1)         ];
    if ([     (2)      ]){
        queue_full(rear);
        return;
    }
    [        (3)         ];
}

/* deleting an item from a queue */
element deleteq(int *front, int rear)
{
    element item;
    if (*front == rear)
        return queue_empty();
    [        (4)         ];
    return [        (5)         ];
}
```