# 1 (5') Stack, Queue and Complexity Analysis

Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 0.5 point if you select a non-empty subset of the correct answers.
*Note that you should write you answers of section 1 in the table below.*

| Question 1 | Question 2 | Question 3 | Question 4 | Question 5 |
|---|---|---|---|---|
| BD | AD | ACD | A | ABC |

**Question 1.** *In the lectures of Week 2, suppose we implement a circular queue by using an array with the index range from* 1 *to* $n$*, then what the size of this queue would be? We assume that the queue is non-empty.*

*(A)* $rear - front + 1$

*(B)* $(rear - front + 1)\%n$

*(C)* $(rear - front + n)\%n$

*(D)* $(rear - front + n)\%n + 1$

**Question 2.** *Which of the following is known to be correct?*

*(A) Stack is a linear data structure and the operations on stacks are more restricted, the same is true for queue.*

*(B) Lists store elements in sequential locations in memory.*

*(C) Both stacks and queues allow us insert or delete an element at the front.*

*(D) We can use two queues to implement stack.*

**Question 3.** *Which of the following is/are applications of queue and stack respectively?*

*(A)* ***Queue****: A resource shared by multiple users/processes;* ***Stack****: Handling function calls*

*(B)* ***Queue****: Loading Balancing;* ***Stack****: Reverse-Polish Notation*

*(C)* ***Queue****: Handling of interrupts in real-time systems;* ***Stack****: Compilers/Word Processors*

*(D)* ***Queue****: IO Buffers;* ***Stack****: Arithmetic expression evaluation*

**Question 4.** *Read the following code, what function does it realize?*

```
void Q4(Queue &Q)
{
   Stack S;
      int d;
      InitStack(S);
      while(!QueueEmpty(Q))
      {
          DeQueue(Q, d)
          Push(S, d);
      }
      while (!StackEmpty(S))
      {
          Pop(S, d);
          EnQueue(Q, d);
      }
}
```

*(A) Use stack to reverse the queue.*

*(B) Use queue to reverse the stack.*

*(C) Use stack to implement the queue.*

*(D) Use queue to implement the stack.*

**Question 5.** *Which of the following comparison is correct?*

*(A)* $n^2 + n^3 = O(n^4)$

*(B)* $\log_2 n = \Theta(\log n)$

*(C)* $\log^2 n = \Omega(\log \log n)$

*(D)* $n! = \omega(n^n)$

## 2 (10') Stack and Queue

**Question 6.** *(2')The following post-fix expression (Reverse-Polish Notation) with single digit operands is evaluated usng a stack:*

$$8\ 2\ 3\ \hat{}\ /\ 2\ 3\ *\ +\ 5\ 1\ *\ -$$

*Note that ˆ is the exponentiation operator. Please write down the corresponding in-fix notation A and the final result:*

$8/2^3 + 2*3 - 5*1$ result: 2

**Question 7.** *(4')***Describe** *how to implement a queue using a singly-linked list. You can use pseudocode or natural language to describe all the operations, especially the key operations.*

```
class Queue {
  private:
    SingleList list;
  public:
    bool isEmpty();
    Type front();
    Void push(Type obj)
    Type pop();
}
```

```
void Queue::isEmpty() {
    return list.isEmpty();
}

Type Queue::front() {
    if(isEmpty())
        throw Error();
    return list.front();
}
```

```
Void Queue::push(Type obj) {
    list.pushBack(obj);
}

Type Queue::pop() {
    if(isEmpty())
        throw Error();
    return list.popFront();
}
```

**Question 8.** ***(1')****If we use an array with size $N$ to implement a normal queue, it gets full when the index* ***Back*** *pointing to the index =* $N-1$

**Question 9.** ***(1')****By implementing the following operations on stack, the value of $x$ is* $a$

*InitStack(st); Push(st,a); Push(st,b); Pop(st,x); Top(st,x);*

**Question 10.** ***(2')****What dose "stack overflow" and "stack underflow" mean? (give a short explanation)*

① "Stack overflow" means the number of elements in the stack has exceed stack's capacity, or someone intends to push elements into stack while the stack is full. There's no room for new elements.

② "Stack underflow" means someone want to access an element of an empty stack or want to pop an element out of an empty stack.

## 3 (8') Complexity Analysis

**Question 11.** ***(3')****Given a fraction of a code as the following, write down the time complexity for each* ***for*** *loop.*

```
for ( i=1; i<n; i*=2) {
   for( j=n; j>0; j/=2) {
            for ( k=j; k<n; k+=2) {
                    sum += (i + j*k)
            }
   }
}
```

$\Theta(n\log^2 n)$

$\Theta(n\log n)$

$\Theta(n)$

**Question 12.** ***(5')*** *Calculate the average processing time $T(n)$ of the following recursive algorithm. Suppose that it takes one unit time for* ***random( int n )*** *to return a random integer which is uniformly distributed in the range [0,n]. Also note that $T(0) = 0$.*
*Hints:The equation $\frac{1}{1*2} + \frac{1}{2*3} + ... + \frac{1}{n*(n+1)} = \frac{n}{n+1}$ might be needed.*

```
int hw( int n) {
  if ( n <= 0 ) return 0;
    else {
            int i = random( n-1 );
            return hw( i ) + hw( n-1-i );
         }
}
```

$$T(n) = \frac{\sum_{i=0}^{n-1}(T(i)+T(n-1-i))}{n} + 1$$

$$= \frac{2\sum_{i=0}^{n-1}T(i)}{n} + 1$$

$$\Rightarrow nT(n) = 2\sum_{i=0}^{n-1}T(i) + n$$

$$(n-1)T(n-1) = 2\sum_{i=0}^{n-2}T(i) + (n-1)$$

$$\Rightarrow nT(n) - (n-1)T(n-1) = 2T(n-1) + 1$$

$$\Rightarrow nT(n) = (n+1)T(n-1) + 1$$

$$\Rightarrow T(n) = n = \Theta(n)$$