Your Name: 陈洋 Your ID: 2018533182HW3 Due date: 11:59 PM, September 29, 2019

# 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.*

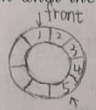
| Question 1 | Question 2 | Question 3 | Question 4 | Question 5 |
|---|---|---|---|---|
| D | AD | ABCD | A | ABC |

C = a/b
r = a - c·b

**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.*

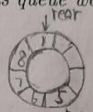
-3/8 = 1 ··· 5

A




5-1+1=5 ✓

B




1-5+1=-3
-3%8=5 ✓

(5-1+8)%8+1=5 D ✓

*(A)* $rear - front + 1$ ✗

*(B)* $(rear - front + 1)\%n$ ✗

*(C)* $(rear - front + n)\%n$ ✗ (+1)

*(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);
    }
}
```

Q6:

8 2 3 ^ / 2 3 * + 5 1 * -

$2^3 = 8$

8/8 = 1  2*3 = 6

1+6 = 7  5*1 = 5

7-5 = 2

8 / 2^3 + 2*3 - 5*1 = 2

(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) $n^2 + n^3 = O(n^4)$ ✓ $\lim_{n\to\infty} \frac{n^2+n^3}{n^4} = 0$

(B) $\log_2 n = \Theta(\log n)$ ✓ $\lim_{n\to\infty} \frac{\log n}{\log n} = 1$

(C) $\log^2 n = \Omega(\log \log n)$ ✓ $\lim_{n\to\infty} \frac{\log^2 n}{\log(\log n)} = \infty$

(D) $n! = \omega(n^n)$ × $\lim_{n\to\infty} \frac{n!}{n^n} = 0$

$\Omega(g(x))$ $O(g(x))$ $\Theta(g(x))$ $o(g(x))$ $\omega(g(x))$

$\lim \frac{f(x)}{g(x)} =$ 0 $0<c<\infty$ $\infty$

$\frac{1}{n!} \sim \frac{e^n}{n\sqrt{2\pi n}}$ $(n\to+\infty)$

## 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 ^ / 2 3 * + 5 1 * -

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

8 / 2^3 + 2*3 - 5*1 = 2 (The process is on the above)

**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.*

First: define a class Single-list, including some functions: bool empty() const ····· determine whether it is empty
Type front() const ····· return the data stored in the first node; void push_back(Type const &) ····· insert a new element at the back of the singly-linked list; Type pop_front() ····· pop the data in the first node.
Then: we can use this singly-linked list to implement a queue.

```
template<typename Type>
class Queue {
    private:
        Single-list list;
    public:
        bool empty() const {
            return the function empty() of the list; };
        Type front() const {
            return the function front() of the list;
        };
```

```
void push(Type const &s)
{ return the function push_back(s) of the list;
};
Type pop()
{ return the function pop_front() of the list;
};
```

The template of the Single_list and the implementations are as follows:

```
template <class T>
class Single_node {
    private:
        T element;
        Single_node *next_node;

    public:
        Single_node( int e, Single_node *n ) :  element(e), next_node(n);
        T retrieve() const;
        Single_node *next() const;
};
```

```
template <class T>
class Single_list {
    private:
        Single_node *list_head;

    public:
        Single_list();
        ~Single_list();

        int size() const;    /* return the length of the list */
        bool empty() const;  /* return true when list is empty */
        T front() const;     /* return the data in the first node */
        T back() const;      /* return the data in the last node */
        Single_node * head() const;  /* return the first node */
        Single_node * tail() const;  /* return the last node */
        int count ( T const & ) const; /* count the number of instances of data */

        void push_front( T const & );  /* insert a node as the first node */
        void push_back( T const & );   /* insert a node as the last node */
        T pop_front();                 /* return the data in the first node and delete the first node */
        int erase( T const & );        /* remove the nodes containing the integer */
};
```

```
template <class T>
class Stack {
    private:
        Single_list list;
    public:
        bool empty() const;
        T top() const;
        void push( T const & );
        T pop();
};

T Single_node :: retrieve() const
{
    return element;
}

Single_node *Single_node :: next() const
{
    return next_node;
}

Single_list :: Single_list : list_head(nullptr)
{
    return;
}

bool Single_list :: empty() const
{
    return ( list_head == nullptr );
}
```

```
bool Single_list :: empty() const
{
    return ( list_head == nullptr );
}

Single_node *Single_list :: head() const
{
    return list_head;
}

template <class T>
T Single_list :: front() const
{
    if ( empty() )
    {
        throw underflow();
    }
    return head() -> retrieve();
}
```

```
template <class T>
void Single_list :: push_front( T const &n )
{
    list_head = new Single_node( n, list_head );
}

template <class T>
int Single_list :: pop_front()
{
    if ( empty() )
    {
        throw underflow();
    }
    int e = front();
    Single_node *ptr = list_head;
    list_head = list_head -> next();
    delete ptr;
    return e;
}
```

```cpp
template <class T>
T Stack :: top() const
{
    if ( empty() )
    {
        throw underflow();
    }
    return list.front();
}

T Stack :: pop()
{
    if ( empty() )
    {
        throw underflow();
    }
    return list.pop_front();
}
```

**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)*

Stack overflow: push(push()) something into the stack when it is full

Stack underflow: pop(pop()) something out of the stack when it is empty (or top())

## 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) {                    O(log²n)
  for( j=n; j>0; j/=2) {                   O(logn)
        for ( k=j; k<n; k+=2) {            O(n)
              sum += (i + j*k)
        }
  }
}
```

**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 );
        }
}
```

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

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

$\Rightarrow hw(n) = n$

$T(n) = 0 + T(i) + T(n-1-i) \quad i \in [0, n-1]$

$nT(n) = n\sum_{m=}^{n-1} T(m) + n$

$\frac{1}{1*2} + \frac{1}{2*3} + \cdots + \frac{1}{n*(n+1)} = \frac{n}{n+1}$

Guess $T(n) = O(n)$ $\exists c$ s.t. $T(n) \le cn$

$T(n) \le 1 + ci + c(n-1-i)$

$= cn + (1-c)$

$\exists\ c \ge 1 \quad T(n) \le cn$

$\therefore T(n) = O(n)$