# CUHK(SZ)-CSC3100 Midterm Exam

2023 Summer

**Note:**
a. No notes or calculators are allowed in the exam.
b. This exam paper has five pages, in double-sided printing.
c. Answer all questions within 100 minutes in an **answer book**.
d. Write your name and student ID on both the exam paper and answer book.

1. (10×2 points) Choose **ONE** solution that best suits each question. (2 points each)

(1). An array is declared as $a[m][n]$ where m is the number of rows while n is the number of columns. Each element of the array occupies b bytes in memory. Denote the address of the first element $a[0][0]$ by "$BA$". What is the address of an element $a[i][j]$ $(0 \leq i < m, 0 \leq j < n)$ if the array is stored row by row (i.e. row-major)?

A. $BA + (i \times n + j) \times b$

B. $BA + (j \times m + i) \times b$

C. $BA + i \times j \times b + m \times n$

D. $BA + m \times n \times b + i \times j$.

(2). What is the **BEST** and **WORST** complexity of quick sort?

A. $O(n)$ and $O(n^2)$

B. $O(\log(n))$ and $O(n^2)$

C. $O(n \log(n))$ and $O(n^2)$

D. $O(n)$ and $O(n \log(n))$

(3). Which statement below is WRONG concerning to stack data structure?

A. push() and pop() are two operations defined in stack.

B. A stack contains a sequence of zero or more items of the same type.

C. List-based stack has no limit on total number of items of the stack.

D. Stack is a First-in-first-out (FIFO) structure.

(4). How many **comparisons** does selection sort make when the input array with size $n$ is already sorted?

A. $O(\log n)$

B. $O(n)$

C. $O(n \log n)$

D. $O(n^2)$

(5). How many **comparisons** does insertion sort make when the input array with size $n$ is already sorted?

A. $O(\log n)$

B. $O(n)$

C. $O(n \log n)$

D. $O(n^2)$

(6). One difference between a queue and a stack is:

A. Queues require linked lists, but stacks do not.

B. Stacks require linked lists, but queues do not.

C. Queues use two ends of the structure; stacks use only one.

D. Stacks use two ends of the structure, queues use only one.

(7). Consider the following pseudo-code:

```
declare a stack of characters
while ( there are more characters in the word to read )
{
   read a character
   push the character on the stack
}
while ( the stack is not empty )
{
   pop a character off the stack
   write the character to the screen
}
```

What is written to the screen for the input “carpets”?

A. serc

B. carpets

C. steprac

D. ccaarrppeettss

(8). What is the output of fun(3) given the following code?

```
static void fun(int n){
   if (n==0) System.out.print("-");
   else{
      System.out.print(n);
      fun(n-1);
      System.out.print(n);
   }
}
```

A. 321-

B. 321-123

C. -123

D. run-time error

(9). What does the function *func* do in general?

```
public static void func(Queue Q)
{
   Stack S = new Stack();
   while (!isEmpty(Q))
   {
      S.push(Q.dequeue());
   }
   while (!isEmpty(S))
   {
      Q.enqueue(S.pop());
   }
}
```

A. Removes the last element from Q

B. Keeps the Q same as it was before the call

C. Makes Q empty

D. Reverses Q

(10). A circular linked list is one in which the last node is connected to the first node. If we set a node `curr` equal to `first` node at the start of traversal, the condition to check if we have reached the end of the list then becomes:

A. `curr==first`

B. `curr.next==null`

C. `curr.next.next==first`

D. `curr.next==first`

2. (5×5 points) Give concise answers to the following short questions (5 points each).

   (a) List five types of common data structures.

   (b) Describe the advantages of a linked list over an array.

   (c) What are the drawbacks of using an array to implement a linear queue?

   (d) What is the primary conceptual difference between a stack and a queue?

   (e) What is the definition of big-oh? Given the mathematical definition for an algorithm with $O(g(n))$ complexity.

3. (5 points) If $T_1(N) = O(f(n))$ and $T_2(N) = O(f(n))$, show if the statement $T_1(N) = O(T_2(N))$ is true or not. Give your proof.

4. (5 points) Let $A$ be a matrix of size $n \times n$, implemented as a 2-dimensional array of double numbers. Consider the following program. What is the expected output?

```
double C = 100;
for (int i=0; i<=n-1; i++) {
    for (int j=0; j<=n-1; j++) {
        double Temp = A[i][j] + C;
        A[i][j] = A[j][i];
        A[j][i] = Temp - C;
    }
}
for (int i=0; i<=n-1; i++) {
    for (int j=0; j<=n-1; j++) {
        System.out.print(A[i][j] + " ");
    }
    System.out.println();
}
```

5. (5 points) In the method F below, q1 and q2 are two queues containing integer items. What should method F print on the screen?

```
public static void F(){
    Queue q1 = new Queue();
    Queue q2 = new Queue();
    for (int i=1; i<10; i=i+2){
        q1.enqueue(i);
        q2.enqueue(i+1);
    }
    while(!q1.isEmpty()){
        System.out.print(q1.dequeue()+" ");
        System.out.print(q2.dequeue()+" ");
    }
}
```

6. (10 points) Given an array of random integers, Push all the zero's of a given array to the end of the array. For example, an array before and after reorganization is like:
Before: `arr[] = 1, 2, 0, 4, 3, 0, 5, 0;`
After: `arr[] = 1, 2, 4, 3, 5, 0, 0, 0;`
Write in Java or pseudo-code. You are recommended to use the following pseudo-code style.

```
pushZero(int arr[]){
    ... // add your operations
}
```

7. (10 points) Given a queue $Q$, you are asked to design an algorithm to **reverse** $Q$ using another queue. You can use standard operations defined in a queue ADT. Write in Java or pseudo-code. You are recommended to use the following pseudo-code style.

```
reverse(queue Q, queue Q_1){ //Q is to be reversed using Q_1.
    ...
    // return either Q or Q_1 that consists of the reversed sequence.
}
```

8. (20 points) Given **two** queues $Q_1$ and $Q_2$ with their standard operations (enqueue, dequeue, isempty, size), implement a stack ADT $S$ with its standard operations (pop, push, isempty, size). Suppose that once $S$ is created, you are given two empty queues $Q_1$ and $Q_2$ by default. Write in Java or pseudo-code. You are recommended to use the following pseudo-code style.

```
class S{
    queue Q_1 //one queue instance
    queue Q_2 //another queue instance

    pop(){
        ... // use Q_1, Q_2 here
    }

    push(x){
        ... // use Q_1, Q_2 here
    }

    ...
}
```