# CS101A DATA STRUCTURE(H)
## QUIZ [2] DS1-5. BASIC DATA STRUCTURES

**NAME:** ____________________ **STUDENT ID:** ____________________

**EMAIL:** ________________________________________

**Start Time:** 20:40 PM **End Time:** 21:00 PM

---

## Q. 1 TRUE OR FALSE (14 pts)

**1.** Starting empty and doubling capacity only when full, there exists a sequence of $m$ `push_backs` where one push costs $\Theta(m)$ while the average cost over all pushes is $O(1)$. ○ **T** ○ **F**

**2.** In a singly linked list with both head and tail pointers, `push_front`, `pop_front`, and `push_back` each run in worst-case $O(1)$ time. ○ **T** ○ **F**

**3.** Given a pointer to a non-head and non-tail node in a singly linked list, deleting that node is a $\Theta(1)$ operation. ○ **T** ○ **F**

**4.** Evaluating the RPN expression `5 1 2 + 4 * + 3 -` yields 14. ○ **T** ○ **F**

**5.** With a hash table of size $m = 7$, hash $h(k) = k \bmod 7$, and *quadratic probing* $h_i(k) = (h(k) + i^2) \bmod 7$, inserting $10, 17, 24, 31$ occupies indices $3, 4, 0, 6$, and inserting 38 then succeeds at index 1. ○ **T** ○ **F**

**6.** $n + \lfloor \log_2 n \rfloor = \Theta(n)$ and $(\log n)^2 = o(n^{0.01})$. ○ **T** ○ **F**

**7.** For fixed constants $A > B > 0$, $(\log n)^A = \omega((\log n)^B)$; moreover, for any $C > 0$, $(\log n)^C = o(n^\varepsilon)$ for every $\varepsilon > 0$. ○ **T** ○ **F**

## Q. 2 ARRAY CAPACITY (8 pts)

Suppose there are two initially empty arrays of capacity 4. You continuously push elements into these arrays. When you want to push an element into a full array, you must increase the array's capacity and copy all the old elements to the new array. The first array's capacity increases by +2 each time. The second array's capacity increases by a factor of 2 each time. Answer the following questions; the subparts are independent.

**(a)** Suppose we insert 7 elements into the **first** array, the unused memory is ____, the total number of copies is _____.

**(b)** Suppose we insert 7 elements into the **second** array, the unused memory is ____, the total number of copies is _____.

**(c)** Suppose we insert 17 elements into the **first** array, the unused memory is ____, the total number of copies is _____.

**(d)** Suppose we insert 17 elements into the **second** array, the unused memory is ____, the total number of copies is _____.

## Q. 3 STACK VIA TWO QUEUES (18 pts)

In the symmetry of data structures lies a quiet elegance: FIFO and LIFO are twin paradigms, they built from the same parts, but with opposite choreography. Building on this duality, revisit the classic trick in reverse: use two FIFO queues `q1` and `q2` to emulate a LIFO stack. Implement `push(x)`, `pop()`, and `top()` by completing the placeholders `/*(1)*/`–`/*(6)*/` in the provided skeleton.

```
class MyStack {
  queue<int> q1, q2;
  void push(int x) {
    /*(1)*/;
    while (/*(2)*/) {
      int v = /*(3)*/;
      /*(4)*/;
    }
    std::swap(q1, q2);
  }
  int top() {
    /*(5)*/;
  }
  int pop() {
    int v = /*(6)*/;
    return v;
  }
  bool empty() const { return q1.empty(); }
};
```

**Fill in the blanks here**

(1) ______________________

(2) ______________________

(3) ______________________

(4) ______________________

(5) ______________________

(6) ______________________

## Q. 4 ASYMPTOTIC ANALYSIS (10 pts)

```
inline void tiny_mix(int& x, int& y, int& z){
  int t = x ^ (y + 0x9e3779b9);
  x = y ^ (z + 0x7f4a7c15);
  y = z ^ (t + 0x85ebca6b);
}
void SolveB(vector<int>& a, int l, int r){
  int n = r - l + 1;
  if(n <= 1) return;
  int c1 = l + (3*n)/5;
  int c2 = l + (4*n)/5;
  int i1 = l, i2 = c1-1, i3 = c2-1;
  if(i1 >= l && i1 <= r && i2 >= l && i2 <= r && i3 >= l && i3 <= r){
    tiny_mix(a[i1], a[i2], a[i3]);
  }
  SolveB(a, l, c1-1);
  SolveB(a, l, c2-1);
}
```

First **write the correct recurrence** for the *number of calls* $T(n)$ made by `SolveB` on an input of size $n$, then **find a function** $g$ such that $T(n) = \Theta(g(n))$. Show your reasoning; only the final $\Theta(\cdot)$ answer is worth 2 points.