CS 201 – Data Structures II (L2), Spring 2024
Quiz # 2

Name: ______________________________________ ID: ________________________

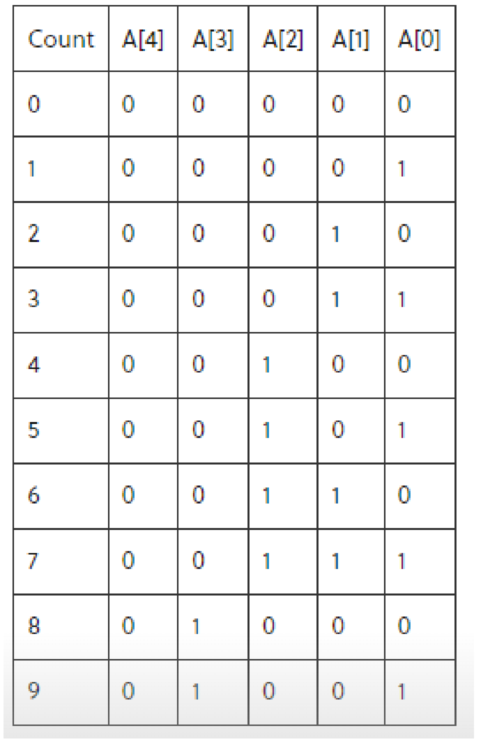
Q1 – You are working with a stack that has an additional functionality of `multipop(k)`. `multipop(k)` will either pop the top *k* elements in the stack, or if it runs out of elements before that, it will pop all of the elements in the stack and stop. The pseudo-code for `multipop(k)` would look like this:

```
multipop(k):
    while stack not empty and k > 0:
        k = k - 1
        stack.pop()
```

a) Given a stack of size n, what is the worst-case complexity of multipop(k) operation in terms of the size of the stack?

b) Compute the amortized cost of multipop(k) **using accounting method.**

Q2 – A binary k-bit counter can be implemented with a k-element binary array that can count up to n. The counter is initially 0. The only operation is increment(A), which adds 1 to the current number in the counter, as shown below:



Count	A[4]	A[3]	A[2]	A[1]	A[0]
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1

```
Increment():
    i = 0
    while i < A.length and A[i] == 1:
        A[i] = 0
        i = i + 1
    if i < A.length:
        A[i] = 1
```

a) What is the worst-case cost per increment?

b) Use **aggregate method** to find the amortized cost per increment?

Q3 - What is Big-O complexity of the given piece of code? Give brief explanation.

a)

```
int i,j,k;

k=1000;

for (i = 1; i <= n/2; i++) {

    for (j = 1; j <= n; j = j * 3) {
        k = k - j;

    }

}
```

b)

```
for (k = 1; k<= 10; k++) {
     for (i = 1; i<=n; i++) {
          for (j = 1; j<=i; j++) {
                 do something in constant time
          }
     }
}
```