**CS 101 Fall 2022 - Quiz 1-B**

**9/20/2022 - 20 Minutes** **Name:** **ID number:**

---

**1. (5 points) Notes of discussion**

I promise that I will complete this QUIZ independently and will not use any electronic products or paper-based materials during the QUIZ, nor will I communicate with other students during this QUIZ.

**True or False: I have read and understood the notes.** √ True ◯ False

**2. (8 points) True or False**

Determine whether the following statements are true or false.

| (a) | (b) | (c) | (d) |
|---|---|---|---|
| F | F | T | T |

(a) (2') Linked list allocates more space (deletion would not reduce this value) than array when we want to frequently insert more elements at the end. Suppose the number of insertions is greater than initial capacity.

(b) (2') Linked list is as efficient as array when we only want to frequently look up elements at the end.

(c) (2') We want to maintain a database which stores students' names and ids and we only import all the data when students get admitted. After that, dropout seldom happens and transferring student never appears. We should use array instead of linked list.

(d) (2') When we want to maintain a task manager where we process each task according to its time when it was created, we could use queue due to its FIFO policy.

**3. (4 points) Stack**

Suppose there is an initially empty stack of capacity 7, and then we do a sequence of 14 operations, which is a permutation of 7 `push` and 7 `pop` operations. If the order of the elements pushed to the stack is `1 2 3 4 5 6 7`, then for each sequence of elements listed below, determine whether it is a possible order of the popped elements. If possible, write down the 14 operations in order.

(a) (2') `7 6 4 5 3 1 2`

**Solution:** Impossible.

(b) (2') `2 4 5 6 3 7 1`

**Solution:** Possible: `push(1), push(2), pop(), push(3), push(4), pop(), push(5), pop(), push(6), pop(), pop(), push(7), pop(), pop()`

**4. (8 points) Reverse-Polish Notation**

Please recall the algorithm we learned in our lecture to evaluate a reverse-Polish notation using a specific data structure. The pseudocode (with some incomplete lines) of this algorithm is given below. Assume the input is guaranteed to be a legal reverse-Polish expression.

**Note:** For example, if the element $x$ is the plus operator $+$, then $a\ x\ b$ in line 6 stands for $a + b$.

```
1:  function EVALUATE-REVERSE-POLISH-NOTATION(expression)
2:      Let S be an initially empty ____(1)____
3:      for each element x in expression do
4:          if x is an operator then
5:              ____(2)____
6:              c ← a x b
7:              Push c to S
8:          else
9:              ____(3)____
10:         end if
11:     end for
12:     return the last element popped from S
13: end function
```

(a) (2') Choose one of the following data structures to fill in the blank (1):

- √ Stack
- ◯ Queue

(b) (2') Choose one of the following statements to fill in the blank (2):

- ◯ Pop $a$ from $S$ and then pop $b$ from $S$
- √ Pop $b$ from $S$ and then pop $a$ from $S$

(c) (2') Fill in the blank (3):

**Solution:** Push $x$ to $S$

(d) (2') Please evaluate the following expression and write down your result directly.

`1 3 5 * + 2 4 * 8 / -`

**Solution:**

$$1 + 3 \times 5 - 2 \times 4/8 = 15.$$