Programming Assignment #1

Lecturer: Prof. Seung-Hwan Baek

Teaching Assistants: Eunsue Choi, Seokjun Choi, Suhyun Shin, Yujin Jeon

***** PLEASE READ THIS GRAY BOX CAREFULLY BEFORE STARTING THE ASSIGNMENT *****

Due date: 11:59PM October 02, 2023

Evaluation policy:

- Late submission penalty
  - 11:59PM October 02 ~ 11:59PM October 03
    - Late submission penalty (30%) will be applied to the total score.
  - After 11:59PM October 04:
    - 100% penalty is applied for that submission.
- Your code will be automatically tested using an evaluation program.
  - Each problem has the maximum score.
  - A score will be assigned based on the behavior of the program.
- We won't accept any submission via email - it will be ignored.
- Do not modify auxiliary files.
  - Such as: utils.h/cpp, evaluate.cpp, and so on.
- Compile your file(s) using repl.it and check your program before the submission.
- Please do not use C++ standard template library.
  - Such as:
    - #include <queue>
    - #include <vector>
    - #include <stack>
  - Any submission using STL library will be disregarded.

File(s) you need to submit:

- pa1.cpp (Do not change the filename!)

Any questions? Please use PLMS - Q&A board.

0. Basic instruction

Please refer to the instruction document, “DataStructure_PA_instructions.pdf”.

```
>> g++ -std=c++11 -o pa1.exe pa1.cpp utils.cpp
```

1. Asymptotic analysis (1 pts)

a. Choose the time complexity of the following **fibonacci** function.

b. `fibonacci`

   Input: An integer n >= 1

   Output: The n-th Fibonacci number, which is an integer.

```
int fibonacci(int n) {
      if(n==1) return 1;
      if(n==2) return 1;

      return fibonacci(n-1) + fibonacci(n-2);
}
```

   1. *O*(log(n))
   2. *O*(nlog(n))
   3. *O*(n^2)
   4. *O*(2^n)

c. Example output: If you choose *O*(log(n)), then print 1

```
>> ./pa1.exe 1
[Task 1]
1
```

2. Asymptotic analysis (1 pts)

a. Choose the time complexity of the following **`fibonacci2`** function.

b. `Fibonacci2`

   Input: An integer n >= 1

   Output: The n-th Fibonacci number, which is an integer

```
int fibonacci2(int n) {
      int f[n+1];
      f[1] = 1;
      f[2] = 1;

      for (int i = 3; i <= n; i++) {
        f[i] = f[i-1] + f[i-2];
      }
      return f[n];
}
```

   1. *O*(1)
   2. *O*(n)
   3. *O*(log(n))
   4. *O*(n^2)

c. Example output: If you choose *O*(1) then print 1

```
>> ./pa1.exe 2
[Task 2]
1
```

3. Application of List (4 pts)

a. Implement a function that can insert or delete an integer into the list. A user can insert an element in ascending order or delete an element at the specific index. If the specified index is out of range of the given list, print “error”.

b. Input & Output

Input: Sequence of commands, which is one of the following,

- `(‘insert’, integer value):` insert integer value at the appropriate position in the list, while ensuring that the elements within the array are always in ascending order. index indicates zero-based index
- `(‘delete’, index):` delete an element at the index in the list. index indicates zero-based index.

Output:

- An array after insertion/deletion in a string separated with the spacebar
- “`error`” if the index is out of range

c. Example input & output

Input	Output
`[(‘insert’,2),(‘insert’,1),(‘insert’,3)]`	`1 2 3`
`[(‘insert’,0),(‘insert’,0),(‘insert’,1)]`	`0 0 1`
`[(‘insert’,0),(‘insert’,1),(‘delete’,0)]`	`1`
`[(‘insert’,0),(‘delete’,1)]`	`error`
`[(‘delete’,0)]`	`error`
`[(‘insert’,5),(‘insert’,9),(‘delete’,0), (‘insert’,1),(‘insert’,2)]`	`1 2 9`

d. Example execution

```
>> ./pa1.exe 3 “[(‘insert’,4),(‘insert’,5),(‘delete’,0),
(‘insert’,1),(‘insert’,0)]”
[Task 3]
```

```
0 1 5
```

4. Matching Parentheses / Stack (3 pts)

a. An important problem in processing arithmetic expression is to make sure their group symbols match up correctly. Arithmetic expressions can contain various pairs of groups symbols, such as

- Parentheses: "(" and ")"
- Braces: "{" and "}"
- Brackets: "[" and "]"

and each opening symbol must correspond to its respective closing symbol.

b. Implement a stack and a function **MatchingParentheses** that returns the matching correctness of arithmetic expression when the expression is given based on your Stack.

c. Input & Output

- Input: An arithmetic expression consists of at least one group of symbols (Parentheses, Braces, Brackets). Please ignore the other input cases
- Output:
  - True : If the given expression satisfies the pair matching
  - False : If the given expression doesn't satisfy the pair matching

d. Example Input & Output

Input	Output
`[()]`	`True`
`()(()){[]}`	`True`
`((){[]}[({}())])`	`True`
`{`	`False`
`()({{})]`	`False`

e. Example execution

```
>> ./pa1.exe 4 “[()]”
[Task 4]
True
```

5. Queue (3 pts)

a. Implement a function that shows results of series of commands for a queue.

b. Command

- `(‘e’,int)`: enqueue integer into the current queue.
- `(‘d’,NULL)`: dequeue one element from the current queue. If dequeue from the empty queue, print only ‘error’.
- `(‘show’,NULL)`: show the value of the current queue. If the queue is empty, print ‘empty’.
- `(‘size’,NULL)`: show the size of the current queue.
- `(‘isEmpty’,NULL)`: show whether the current queue is empty or not. If the queue is empty, print ‘T’. Otherwise, print ‘F’.
- `(‘clear’,NULL)`: dequeue all elements from the current queue.

c. Example Input & Output

Input	Output
`[(‘e’,4),(‘e’,7),(‘d’,NULL),(‘show’,NULL)]`	`7`
`[(‘e’,5),(‘e’,2),(‘d’,NULL),(‘size’,NULL)]`	`1`
`[(‘d’,NULL),(‘isEmpty’,NULL)]`	`error`
`[(‘e’,5),(‘e’,2),(‘d’,NULL),(‘e’,1),(‘size’,NULL),(‘isEmpty’, NULL),(‘show’,NULL)]`	`2 F 2 1`
`[(‘e’,4),(‘e’,7),(‘e’,5),(‘isEmpty’,NULL),(‘clear’,NULL),(‘isEmpty’,NULL), (‘show’,NULL)]`	`F T empty`

d. Example execution

```
>> ./pa1.exe 5 “[(‘e’,4),(‘e’,7),(‘d’,NULL),(‘show’,NULL)]”
[Task 5]
7
```

6. Circular Queue (3 pts)

a. Implement a function that shows the values in a circular queue with a counter. If “e” is called for an already full queue or the “d” operation is called for an empty queue, there should be no changes to the queue. The maximum number of elements (n) in the queue is five.

b. Input & Output

Input: Sequence of commands, which is one of the following,

- (‘e’, int): enqueue integer into the current queue
- (‘d’): dequeue from the current queue
- (‘show’): show the value of the current queue. If the queue is empty, print ‘empty’
- (‘size’): show the number of elements in the current circular queue.

Output

- Values in the circular queue (mod size n = 5), from the front to the rear. String separated with the spacebar
- No pointer movement if dequeue applied on an empty queue or enqueue applied on a full queue

c. Example input & output

Input	Output
[(‘e’,5),(‘e’,3),(‘d’, NULL),(‘show’, NULL)]	3
[(‘e’,5),(‘e’,3),(‘d’, NULL),(‘e’,6),(‘size’, NULL)]	2
[(‘e’,3),(‘d’, NULL),(‘show’, NULL)]	empty
[(‘e’,5),(‘d’, NULL),(‘e’,3), (‘e’,6),(‘e’,9),(‘e’,1), (‘d’, NULL),(‘d’, NULL), (‘e’,7),(‘e’,2),(‘show’, NULL)]	9 1 7 2
[(‘e’, 1),(‘e’, 2),(‘e’, 3),(‘e’, 4),(‘e’,5),(‘e’, 6),(‘show’, NULL)]	1 2 3 4 5

d. Example execution

```
>> ./pa1.exe 6 “[(‘e’,5),(‘e’,3),(‘d’, NULL),(‘e’,6),(‘size’, NULL)]”
[Task 6]
2
```