Programming Assignment #1

Lecturer: Prof. Jaesik Park

Teaching Assistants: Hyeonjun Lee, Hyunsoo Ahn, Sangwoo Ryu

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

Due date: 11:59PM March 18, 2022

Evaluation policy:

- Late submission penalty
  - 11:59PM March 18 ~ 11:59PM March 19
    - Late submission penalty (30%) will be applied to the total score
  - After 11:59PM March 19:
    - 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
- Please do not use C++ standard template library
  - Such as:
    - #include <queue>
    - #include <vector>
    - #include <stack>
  - Any submission using STL library will be disregarded

Any questions?

- Please use LMS - Q&A board

0. Basic instruction

a. Please refer to the attached file named PA_instructions.pdf

1. Asymptotic analysis (1 pts)

a. Choose the TIGHT bound of the following **`maxProduct`** function

b. `maxProduct`
   Input: An integer n >= 1, arrays A & B storing n integers
   Output: The maximum product of elements from each A and B

```
int maxProduct(int n, int* A, int* B) {
    int currMax = 0;
    for (int i = 1; i < n; i++)
        for (int j = 1; j < n; j++)
            if (currMax < A[i]*B[j])
                currMax = A[i]*B[j];
    return currMax;
}
```

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

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

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

2. Asymptotic analysis (1 pts)

a. Choose the TIGHT bound of the following **medianSearch** function

b. `medianSearch`
Input: An integer n >= 2, an ascending sorted array A storing n real numbers
Output: An array B which contains nth, n/2th, n/4th, … elements of A

```
double* medianSearch(int n, double* A) {
    double *B = new double[n];
    /* B is allocated as same size as A */
    int j = 0;
    for (int i = n; i >= 1; i = i/2) {
        B[j] = A[i-1];
        j++;
    }
    return B;
}
```

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

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

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

3. List (4 pts)

a. Implement a function that can insert or delete an integer into the ascending order “sorted” list. An user can delete a specified smallest element. If the specified element is out of range of the given list, print “error”

Tips: Please do not try to implement sorting algorithm for this task

b. Input & Output

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

- `(‘insert’,integer)`: insert `integer` into the appropriate position in the sorted list.
- `(‘delete_at’,i)`: delete an item that is `i`-th smallest element in the list. `i` 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’,1),(‘insert’,2)]`	`1 2`
`[(‘insert’,2),(‘insert’,1),(‘insert’,3)]`	`1 2 3`
`[(‘insert’,1),(‘insert’,3),(‘delete_at’,1)]`	`1`
`[(‘insert’,1),(‘delete_at’,2),(‘insert’,2)]`	`error`

`[(‘insert’,1),(‘insert’,3),(‘insert’,2),`
`(‘delete_at’,1)]`
	`1 3`

d. Example execution

```
>> ./pa1.exe 3 “[(‘insert’,1),(‘insert’,3),(‘delete_at’,1),
(‘insert’,2)]”
[Task 3]
1 2
```

4. Stack (3 pts)

a. Implement a function that prints the top values of the stack when “top” operation is called after the sequence of “**push**” or “**pop**” operations. If the stack is empty, and the **“top”** operation is called, then print “-1”, If “pop” operation from the empty stack then print “**error**”

b. Input & Output

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

- (‘push’,integer): push integer into the current stack (integer is always positive)
- (‘pop’,NULL): pop the top value of the current stack (this operation will print nothing)
- (‘top’,NULL): print the top value of the current stack (print ‘-1’ if the current stack is empty)

Output:

- Expected printed values after processing the whole sequence, in a string separated with the spacebar
- “error” if the pop operation is executed on an empty stack

c. Example Input & Output

Input	Output
[(‘push’,5),(‘push’,3),(‘top’,NULL)]	3
[(‘push’,3),(‘top’,NULL),(‘pop’,NULL),(‘push’,5),(‘top’,NULL)]	3 5
[(‘push’,5),(‘pop’,NULL),(‘top’,NULL)]	-1
[(‘pop’,NULL)]	error
[(‘pop’,NULL),(‘push’,5),(‘top’,NULL)]	error

d. Example execution

```
>> ./pa1.exe 4 “[(‘push’,3),(‘top’,NULL),(‘pop’,NULL),
(‘push’,5),(‘top’,NULL)]”
[Task 4]
3 5
```

5. Queue (3 pts)

a. Implement a function which shows the value in the queue from the head to tail.

b. Input & Output

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

- (‘enqueue’,integer): enqueue integer into the current queue

Output:

- Values in the queue from the head to the tail, in a string separated with the spacebar

c. Example Input & Output

Input	Output
[(‘enqueue’,5)]	5
[(‘enqueue’,5),(‘enqueue’,2)]	5 2
[(‘enqueue’,5),(‘enqueue’,2),(‘enqueue’,3)]	5 2 3

d. Example execution

```
>> ./pa1.exe 5 “[(‘enqueue’,5),(‘enqueue’,2),(‘enqueue’,3)]”
[Task 5]
5 2 3
```

6. Queue (3 pts)

a. Implement a function that shows the value of a queue after the sequence of arbitrary queue operations. If the queue after the operations is empty, print “**`empty`**”. If “dequeue” operates on an empty queue, print “**`error`**”.

b. Input & Output

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

- `(‘enqueue’,integer)`: enqueue `integer` into the current queue
- `(‘dequeue’,NULL)`: dequeue from the current queue

Output

- Values in the queue from the head to the tail, in a string separated with the spacebar
- “`empty`” if the queue is empty
- “`error`” if the “dequeue” operation is executed on an empty queue

c. Example input & output

Input	Output
[(‘enqueue’,5),(‘enqueue’,3),(‘dequeue’,NULL)]	3
[(‘enqueue’,5),(‘enqueue’,3),(‘dequeue’,NULL),(‘enqueue’,5)]	3 5
[(‘enqueue’,3),(‘dequeue’,NULL)]	empty
[(‘enqueue’,5),(‘dequeue’,NULL),(‘dequeue’,NULL)]	error
[(‘enqueue’,5),(‘dequeue’,NULL),(‘dequeue’,NULL),(‘enqueue’,3)]	error

d. Example execution

```
>> ./pa1.exe 6 “[(‘enqueue’,5),(‘enqueue’,3),(‘dequeue’,NULL)]”
[Task 6]
3
```