**Programming Assignment #1**

Lecturer: Prof. Jaesik ParkTeaching Assistants: Seunguk Do, Suhyeon Jeong, Minjae Jeong, Sangjun Lee

|  |
| --- |
| *\*\*\*\* PLEASE READ THIS GRAY BOX CAREFULLY BEFORE STARTING THE ASSIGNMENT \*\*\*\**  Due date: 11:59PM March 21, 2023  Evaluation policy:   * Late submission penalty   + 11:59PM March 21 ~ 11:59PM March 22     - Late submission penalty (30%) will be applied to the total score.   + After 11:59PM March 22:     - 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. * Never try to copy your classmates’ source code. Both of you and your classmates will get severe penalty, and the case will be reported to the University’s office.   File(s) you need to submit:   * pa1.cpp (Do not change the filename!)   Any questions? Please use PLMS - Q&A board. |

1. 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)
2. Choose the TIGHT bound of the following arrayInsert function.
3. arrayInsert  
   Input arguments: An integer n >= 1, an array A storing n integers, inserting position pos, inserting element x

|  |
| --- |
| void arrayInsert(int n, int\* A, int x, int pos) {  n++;  for (int i = n; i >= pos; i--)  A[i] = A[i-1];  A[pos – 1] = x;  return;  } |

1. *O*(1)
2. *O*(n)
3. *O*(n log (n))
4. *O*(n^2)
5. Example output: If you choose *O*(1), then print 1

|  |
| --- |
| >> ./pa1.exe 1  [Task 1]  1 |

1. Asymptotic analysis (1 pts)
2. Choose the TIGHT bound of the following maxSubarraySum function
3. maxSubarraySum  
   Input: An integer n >= 2, an array A storing n integer numbers  
   Output: Define subarray as { A[i], A[i+1], …, A[j] } for all integer i <= j < n. Find and return the maximum value of sum of subarray of A.

|  |
| --- |
| int maxSubarraySum(int n, int \*A) {  int max\_sum = 0;  for (int i = 0; i < n; i++) {  for (int j = i; j < n; j++) {  int cur\_sum = 0;  for (int k = i; k <= j; k++) {  cur\_sum += A[k];  }  if (cur\_sum > max\_sum){  max\_sum = cur\_sum;  }  }  }  return max\_sum;  } |

1. *O*(n log (n))
2. *O*(n)
3. *O*(n^2)
4. *O*(n^3)
5. Example output: If you choose *O*(n log (n)), then print 1

|  |
| --- |
| >> ./pa1.exe 2  [Task 2]  1 |

1. List (4 pts)
2. Implement a function that can insert or replace an integer in the list. A user can insert or replace an element at the specific index. If the specified index is out of range of the given list, print “error”.
3. Input & Output

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

* (‘insert’,i): insert i into the i-th index in the list. i-th index indicates zero-based index.
* (‘replace\_at’,index): replace an element at the index to 0. 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

1. Example input & output

|  |  |
| --- | --- |
| Input | Output |
| [(‘insert’,0),(‘insert’,1)] | 0 1 |
| [(‘insert’,0),(‘insert’,0),(‘insert’,2)] | 0 0 2 |
| [(‘insert’,0),(‘insert’,0),(‘insert’,1)] | 0 1 0 |
| [(‘insert’,1)] | error |
| [(‘insert’,0),(‘insert’,1),(‘replace\_at’,2)] | error |
| [(‘insert’,0),(‘insert’,1),(‘insert’,1), (‘replace\_at’,2)] | 0 1 0 |

1. Example execution

|  |
| --- |
| >> ./pa1.exe 3 “[(‘insert’,0),(‘insert’,1),(‘insert’,1), (‘replace\_at’,2)]”  [Task 3]  0 1 0 |

1. Stack (3 pts)
2. 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 **“**top**”** operation is called for an empty stack, print “-1”, If “pop” operation is called for an empty stack, print “error” and the program terminates. The stack can contain positive integers.
3. Input & Output

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

* (‘push’,integer): push integer into the current stack, if integer is negative, skip the push step
* (‘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, and then the program terminates.

1. Example Input & Output

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

1. Example execution

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

1. Circular Queue (3 pts)
   1. Implement a function that shows the values in a circular queue with a Boolean indicator. If the queue after the operations is empty, print “empty”. If “enqueue” operates on a full queue or if “dequeue” operates on an empty queue, print “error”. The maximum number of elements (n) in the queue is four.
   2. 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 circular queue (mod size n = 4), from the front to the rear. String separated with the spacebar, empty memory shows NULL
* “empty” if the queue is empty when the operations are terminated
* “error” if the “dequeue” operation is executed on an empty queue or if the “enqueue” operation is executed on a full queue
  1. Example input & output

|  |  |
| --- | --- |
| Input | Output |
| [(‘enqueue’,2),(‘dequeue’,NULL), (‘enqueue’,7)],(‘enqueue’,4)] | 7 4 |
| [(‘enqueue’,3),(‘dequeue’,NULL)] | empty |
| [(‘enqueue’,1),(‘enqueue’,4),(‘enqueue’,3), (‘enqueue’,2),(‘enqueue’,5)] | error |
| [(‘enqueue’,5),(‘dequeue’,NULL),(‘dequeue’,NULL)] | error |
| [(‘enqueue’,5),(‘enqueue’,3),(‘dequeue’,NULL), (‘enqueue’,6),(‘enqueue’,9),(‘enqueue’,1),(‘dequeue’,NULL),(‘dequeue’,NULL),(‘enqueue’,7),(‘enqueue’,2)] | 9 1 7 2 |

* 1. Example execution

|  |
| --- |
| >> ./pa1.exe 6 “[(‘enqueue’,2),(‘dequeue’,NULL),(‘enqueue’,7),(‘enqueue’,4)]”  [Task 6]  7 4 |

1. Double-Ended Queue (3 pts)  
   1. Implement a function that shows the values in a double-ended queue with a counter. If “insert” is called for an already full queue or the “erase” 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 four.
   2. Input & Output

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

* (‘insert\_front’,integer): insert integer into the beginning of the queue.
* (‘insert\_back’,integer): insert integer into the end of the queue.
* (‘erase\_front’,NULL): erase the first element the queue.
* (‘erase\_back’,NULL): erase the last element the queue.

Output

* Values in the dequeue (mod size n = 4), from the beginning to the end. String separated with the spacebar, empty memory shows “empty”
* No pointer movement if dequeue applied on an empty queue or enqueue applied on a full queue.
  1. Example input & output

|  |  |
| --- | --- |
| Input | Output |
| [(‘insert\_front’,5),(‘ insert\_front’,3),(‘erase\_back’,NULL)] | 3 |
| [(‘insert\_front’,5),(‘ insert\_back’’,3),(‘erase\_front’,NULL),(‘insert\_front,6)] | 6 3 |
| [(‘insert\_front’,3),(‘erase\_back’,NULL)] | empty |
| [(‘insert\_front’,5),(‘erase\_front’,NULL),  (‘insert\_front’,3),(‘insert\_back’,6),  (‘insert\_front’,9),(‘insert\_front’,1),  (‘erase\_back’,NULL),(‘erase\_back’,NULL),  (‘insert\_back’,7),(‘insert\_back’,2)] | 1 9 7 2 |

* 1. Example execution

|  |
| --- |
| >> ./pa1.exe 6 “[(‘insert\_front’,5),(‘insert\_front’,3),(‘erase\_back’,NULL),  (‘insert\_front’,6)]”  [Task 6]  6 3 |