**Tutorial 4**

1. Suppose that *S* is a stack. List the content of the stack after each operation and show the output value if a value is returned from the operation.

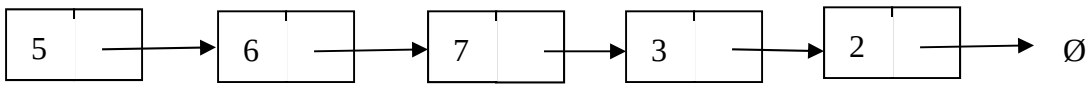
| Operation | Output | Bottom – Stack – Top |
|---|---|---|
| S.stack_init( ) | | |
| S.empty( ) | | |
| S.push(8) | | |
| S.push(-5 ) | | |
| S.pop( ) | | |
| S.push(2) | | |
| S.top( ) | | |
| S.pop( ) | | |
| S.empty( ) | | |
| S.top( ) | | |

2. Suppose that *start* is a reference to the first node of a singly-linked list. Write an algorithm that is passed *start* and a value *val*. The algorithm adds a node to the end of the linked list whose data field is *val*. What is the worst case time complexity of your algorithm?

3. A pointer *start* points to the first element of a doubly-linked list *L*. Write an algorithm that deletes the smallest element in *L*.

4. Using the operations *front()*, *enqueue(val)* and *dequeue()*, write the pseudo-code of a recursive algorithm to append a queue *P* (which may be empty) onto the end of another queue *Q*, leaving *P* empty.

5. The pointer *start* points to the first element of a singly-linked list *L*. Write a recursive algorithm to return a reference to the first element that has a value that is greater than the next element in *L*. If no such element exists, return null. For example if the linked list is the following




then the algorithm will return a reference to the third element (which has the value "7").