[redacted] [redacted] [redacted] [redacted]

1. What is the result of executing the following code snippet?

Assume all required libraries are included and no compile-time/runtime errors occur.

```
int main() {
   list<int> myList;
   for (int i=1; i<6; i++)
      myList.push_back(i);

   for (list<int>::iterator it = myList.begin(); it != myList.end(); it++ )
      *it = *it - 2;

   for (list<int>::iterator it = myList.begin(); it != myList.end(); it++ )
      cout << *it << " ";

   return 0;
}
```

A. [Correct Answer] `-1 0 1 2 3`
B. None of the other options is correct.
C. `1 2 3 4 5`
D. [Your Answer] `1 2 3 4`
E. `-1 0 1 2`

2. We have implemented the Stack ADT as an array. Every time the array is full, you resize the array creating a new array that can hold 3 elements more than the previous array and copy values over from the old array. What is the total running time for n pushes to the stack.

A. [Correct Answer] [Your Answer] $O(n^2)$.
B. $1/3 * O(n)$.
C. $O(n)$.
D. $O(1)$.
E. $O(\log n)$.

3. In implementing Queue ADT, using which of the following data structure gives best asymptotic runtime for `enqueue` and `dequeue`? (Assume best possible implementation for queue using provided data structure)

A. Doubly linked list with head pointer only.
B. [Your Answer] Singly linked list with head and tail pointer.
C. Doubly linked list with head and tail pointer.
D. [Correct Answer] Exactly two of the other options are correct.
E. Singly linked list with head pointer only.

4. Suppose we have implemented the Stack ADT as a singly-linked-list with `head` and `tail` pointers and no sentinels. Which of the following best describe the running times for the functions `push`and `pop`, assuming there are $O(n)$ items in the list, and that the top of the Stack is at the `head` of the list?

A. $O(n)$ for `push` and $O(1)$ for `pop`.
B. None of the options is correct
C. $O(1)$ for `push` and $O(n)$ for `pop`.
D. [Correct Answer] [Your Answer] $O(1)$ for both.
E. $O(n)$ for both.

5. Suppose `queue<int> q` contains 6 elements `1, 2, 3, 4, 5, 6` (enqueued in that order). What is the result of executing the following code snippet? (Assume member function `front()` returns the value found at the front of the queue without removing it.)

```
for(int i = 1; i < 7; i++) {
    if(i % 2 == 1) {
        q.enqueue(q.front());
        q.dequeue();
    }
}
```

A. even numbers in q are reversed.
B. [Correct Answer] [Your Answer] elements in the front half of the original q are now in the back half.
C. the front half of q contains even elements and the back half of q contains odd elements.
D. odd numbers in q are reversed.
E. q remains the same.