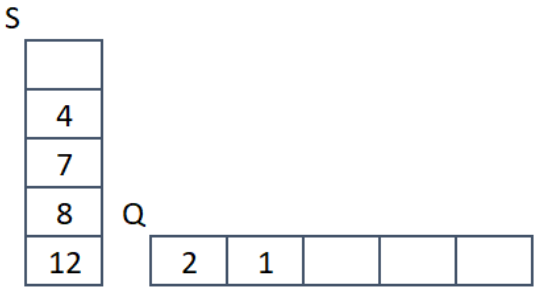
CS210 Data Structure & Algorithms (181) Major Exam 1 Name_____SOLUTION_____ ID_____
Weight: 15% Time: 45 minutes

Solve all Questions

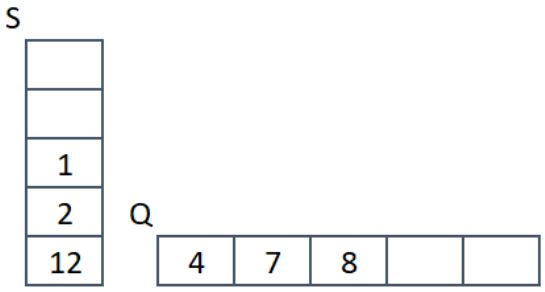
Q1 [2 points] Give the Big-Oh notation for the following functions

$f(N) = N^2 + \log N^2 + 2N \log N$	**$O(N^2)$**
$f(N) = (N \,.\, (100N + 5000000))^{\,2}$	**$O(N^4)$**
$f(N) = N^{1/2} + \log (\log N)$	**$O(N)$**
$f(N) = 1000^{100} + \log N$	**$O(\log N)$**

Q2. [2 points] Assume you have a non-empty Stack S and a Queue Q of type integers. Provide a sequence of calls to modify the contents of S and Q from scenario_1 to scenario_2.



scenario_1



scenario_2

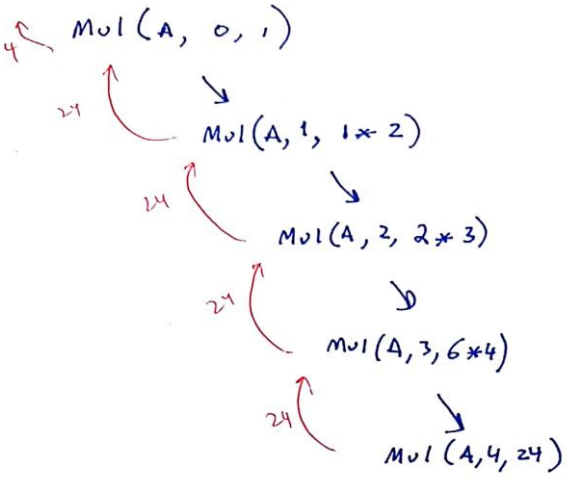
```
Q.enqueue(S.pop())
Q.enqueue(S.pop())
Q.enqueue(S.pop())
S.push(Q.dequeue())
S.push(Q.dequeue())
```

Q3 [4]. Given the following code, estimate the number of operations and describe the worst case running time in Big-Oh notation in terms of the variable n. Show trace where appropriate.

```
public int Mul(int [] A, int i, int t)
{
   if (i >= A.length-1)
     return t;
   else
     return Mul(A, ++i, t * A[i]);
}
```

Mul() is recursively called i times. I would run up to n, which is the length of this array. The run time will be O(n)

Mul(new int[] {1,2,3,4}, 0, 1)




```
public void diag(int n) {
  int count = 0;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < i; j++)
      if(i==j)
         count++;
}
```

The if statement will execute n/2 * (n+1) times

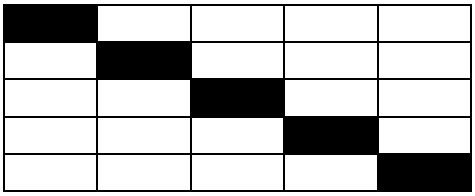
1+2+3+4+5+… = $\frac{n(n+1)}{2}$

which is O(n^2)

diag(5)

```
i:0 (1 times)
j=0

i:1 (2 times)
j=0
j=1

i:2 (3 times)
j=0
j=1
j=2

i:3 (4 times)
j=0
j=1
j=2
j=3

i:4 (5 times)
j=0
j=1
j=2
j=3
j=4
```

Q4. [3 points] Write an iterative method **public static double addDiagonal(double [][] D)** that adds and returns all the left-diagonal values of a n x n 2-dimensional array.



Show a trace to determine the runtime function T(n) estimating the number of operations.

```
public static double addDiagonal(double [][] D)
{
  double total = 0;
  for(int i=0;i<D.length;i++)
     total+=A[i][i];
  return total;
}
```

We must note that the size of the matrix is n x n, which means it is a square matrix. We estimate the number of operation to be:

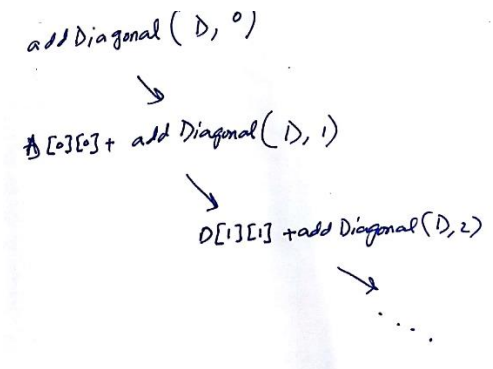
T(n) = 5n + 3

which is O(n)

[+1 Bonus] Write the above method as a recursive method. Estimate the runtime of this method.

```
public static double addDiagonal(double [][] D, int i)
{
  if(i>=D.length)
    return 0;
  else
    return D[i][i] + addDiagonal(D, ++i);
}
```

We can estimate that the run time T(n) = 5c n which is O(n)



Q5. [4 points] Modify the method insert that you defined in the double linked list class DList. The modified method has the following definition.

public static void insert2(Node N, int pos)

This method inserts a node *N* at position *pos* in the DList. You may start counting at the head. Make sure your method checks if value of *pos* is valid (i.e. *pos* must be a non-negative integer with value less than size).

```
public static void insert2(Node N, int pos){
  //we assume the first position is zero
  if(pos<0 || pos > size+1)
    return;
  if(pos==0)//add to the beginning
  {
    N.next = Head;
    N.prev = null;
    Head.prev = N;
    Head = N;
  }
  else if(pos == size) // we add at the end
  {
    N.next = null;
    N.prev = Tail;
    Tail.next =N;
    Tail = N;
  }
  else
  {
    //we add in the middle
    Node temp = Head;
    for(int i=1;i<pos;i++)
      temp = temp.next;

    N.next = temp.next;
    N.prev = temp.prev;
    temp.next = N;
    N.next.prev = N;
  }
  size++;
  return;
}
```