CMPS 12B
Introduction to Data Structures
Quiz 3 Review Problems

1. Given classes `Node` and `NodeTest` defined below, answer the following questions.
   a. Draw a picture of the linked data structure at point (a) in function `main()` of `NodeTest.java`.
   b. Trace execution of `main()` up to point (b) and write the output as it would appear on the screen.
   c. Write instructions that will insert a new Node with item value 4 into position 3 of the list, i.e. insert the new `Node` between the 7 and the 5.

```
// file: Node.java
public class Node{
   // fields
   public int item;
   public Node next;
   // constructor
   public Node(int x){
      item = x;
      next = null;
   }
}

// file: NodeTest.java
public class NodeTest{
   public static void main(String[] args){
      Node H = new Node(9);
      H.next = new Node(7);
      H.next.next = new Node(5);
      // part (a) refers to this point in the code

      for(Node N=H; N!=null; N=N.next) System.out.print(N.item+" ");
      System.out.println();
      // part (b) refers to this point in the code

      // part (c) refers to this point in the code
      // your code goes here
  }
}
```

2. Given the Node class in the previous problem and a linked list based on that class, fill in the function definitions below.

   a. Write a recursive function called `printForward()` that prints out the items from head to tail.

```
static void printForward(Node H){
   // your code goes here
}
```

   b. Write a recursive function called `printBackward()` that prints out the items from tail to head.

```
static void printBackward(Node H){
   // your code goes here
}
```

3. Write a *recursive* Java function called `product()` which, given a head reference to a linked list based on the Node class defined below, returns the product of the items in the list. The product of an empty list is defined to be 1.

```
class Node{
   int item;
   Node next;
   Node(int x){
      item = x;
      next = null;
   }
}

// In some class in the same package as Node:

static int product(Node H){
   // Your code goes here
}
```

4. Write functions push() and pop() for the Java implementation of an integer stack outlined below. The stack is implemented as a singly linked list with a top Node reference. Function push() inserts a new item onto the top of the stack by inserting a new Node at the head of the list. Function pop() deletes the top item, and returns its value.

```
class Stack{
   private class Node{
      int item;
      Node next;
      Node(int item){
         this.item = item;
         this.next = null;
      }
   }
   private Node top;
   private int numItems;
   public Stack(){top = null; numItems = 0;}

   void push(int x){
      // your code goes here
   }

   int pop(){
      // your code goes here
   }
   // other Stack methods would follow
}
```

5. Trace the following C program and show output *exactly* as it would appear on the screen.

```
#include<stdio.h>
#include<stdlib.h>

int f(int x, int y){
   int u;
   u = x*y;
   printf("in f\n");
   return( x+u+y );
}

int g(int* p, int* q){
   int v;
   v = *p + *q;
   printf("in g, before f\n");
   *q = f(v, *p);
   printf("in g, after f\n");
   return( v-*q );
}

int main(void){
   int a=1, b=2, c=3;
   printf("in main, before f and g\n");
   a = f(a, b);
   b = g(&b, &c);
   printf("in main, after f and g\n");
   printf("a = %d, b = %d, c = %d\n", a, b, c);
   return(EXIT_SUCCESS);
}
```