

CS 326-01-09S
OBJECT ORIENTED DESIGN
SUMMER 2009

Program 2

As a continuation of Program 1, you will now use objects (in a somewhat limited way) to solve the exact same problem as before: that knight path problem.

The use of recursion in this project is forbidden. Instead, you will use manual stack manipulation to implement depth first search.

Use the following classes in your code. You do not have to use these names exactly. Put each class in its own file. All data fields must be private. Methods may be public or private as appropriate. You may augment the classes below with whatever additional data fields or methods you find appropriate, but the classes must have the minimum attributes described below.

```
class Position {
```

should have row and column variables to indicate exactly where on the board you are.

```
}
```

```
class LinkedListNode {
```

should have an Object variable to store information and a next pointer to get to the next element in the linked list.

```
}
```

```
class LinkedList {
```

should control the entire linked list structure. Should contain a head and tail pointer to the appropriate linked list nodes. You should have a method that adds an Object to the head of the linked list, one that removes and returns an Object from the head of the linked list, and one that adds an Object to the tail of the linked list. (You do not have to write a method that removes an element from the tail of the linked list; that one is a lot nastier, and we will never need that functionality anyway.) You should also write a method that determines whether the linked list is empty, a total of four methods at minimum.

```
}
```

```
class Stack {
```

should control a stack by means of a linked list. You should write a push method that adds an Object to the stack, a pop method that removes and returns an Object from the stack, and a

method to determine whether the stack is empty, a total of three methods at minimum. You do not have to worry about what happens when you try to remove an element from an empty stack. If you write this program correctly, you will never pop items from an empty stack anyway. Your Stack class should use your LinkedList class described above; however, your Stack class should make no direct reference to your LinkedListNode class.

```
}
```

```
class PG2 {
```

This class must be called exactly this. It is this class that the autograder will attempt to execute. Your main method must be in this class. As mentioned above, this class should solve the knight path problem without using recursion. You may freely use the Stack class described above, but you may not directly use your LinkedList and LinkedListNode classes.

```
}
```

Create a directory called “PG2” in the root your shared folder. Place all files pertaining to this program in this directory. Place a (possibly empty) file called “DONE” in this folder when you are ready to have it graded.