Question 1 (10 pts)

a) Create a control flowgraph for the sieve algorithm. To the left of the line numbers in the source code clearly identify the nodes that will be used in your graph. Once you have identified the nodes, draw the control graph. (4pts)

```c
/* Find all primes from 2-upper_bound using Sieve of Eratosthanes */

#include

typedef struct IntList {
    int value;
    struct IntList *next;
} *INTLIST, INTCELL;

INTLIST sieve ( int upper_bound ) {
    INTLIST prime_list = NULL; /* list of primes found */
    INTLIST cursor; /* cursor into prime list */
    int candidate; /* a candidate prime number */
    int is_prime; /* flag: 1=prime, 0=not prime */

    /* try all numbers up to upper_bound */
    for (candidate=2; candidate <= upper_bound; candidate++) {
        is_prime = 1; /* assume candidate is prime */
        for(cursor = prime_list; cursor; cursor = cursor->next) {
            if (candidate % cursor->value == 0) {
                /* candidate divisible by prime */
                /* in list, can't be prime */
                is_prime = 0;
                break; /* "for cursor" loop */
            }
        }
        if(is_prime) {
            /* add candidate to front of list */
            cursor = (INTLIST) malloc(sizeof(INTCELL));
            cursor->value = candidate;
            cursor->next = prime_list;
            prime_list = cursor;
        }
    }
    return prime_list;
}
```
b) Provide a set of test cases that would give 100% Node Coverage (NC). (2pts)
c) Provide a set of test cases that would give 100% Edge Coverage (EC). (2pts)
d) Is 100% NC or 100% EC possible in general? Why, or why not? (2pts)

Question 2 (10 pts)

Write the formal specification for a stack. Be specific and provide types, state, data invariants, and operations.

Question 3 (9 pts)

Given the following program:

```java
1: public int fibonacci (int i) {
2:     int fib1 = 1; // fib(n-1)
3:     int fib2 = 1; // fib(n-2)
4:     int fib = 0;
5:     int j;
6:     if (i <= 1)
7:         fib = 1;
8:     else
9:         for (j=1; j<i;
10:            j++)
11:             fib = fib2 + fib1 ;
12:             fib2 = fib1 ;
13:             fib1 = fib ;
14:     return fib ;
}
```

Give test cases that will kill the following mutations (3pts each):

(a) Line 6: if (i < 1)
(b) Line 6: if (i == 1)
(c) Line 12: fib2 = fib;

Question 4 (11 pts)

Find a design pattern that is not described in our books.

1. Draw the UML class diagram (4 pts)
2. Draw a UML sequence diagram that exemplifies its operation (4 pts)
3. Where/why would you use this design pattern? (3pts)