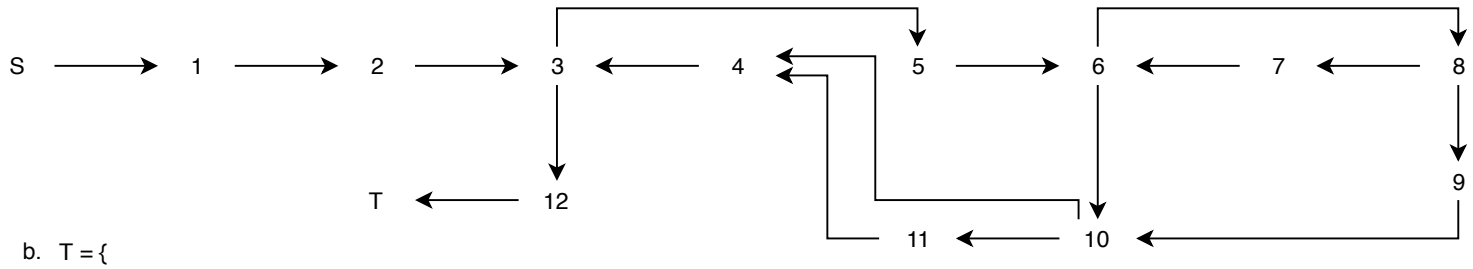
Alex Ackerlund
Ryan Cummings
Bruce Clark

1.a.

```
1.  /* Find all primes from 2-upper_bound using Sieve of Eratosthanes */
2.
3.  #include
4.  typedef struct IntList {
5.            int value;
6.            struct IntList *next;
7.            } *INTLIST, INTCELL;
8.  INTLIST sieve ( int upper_bound ) {
9.
10.        INTLIST prime_list = NULL;   /* list of primes found */
11.        INTLIST cursor;              /* cursor into prime list */
12.        int candidate;               /* a candidate prime number */
13.        int is_prime;                /* flag: 1=prime, 0=not prime */
14.
15.         /* try all numbers up to upper_bound */
16.         for (candidate=2;
17.
18.              candidate <= upper_bound;
19.              candidate++) {
20.
21.          is_prime = 1; /* assume candidate is prime */
22.          for(cursor = prime_list;
23.
24.               cursor;
25.               cursor = cursor->next) {
26.
27.             if (candidate % cursor->value == 0) {
28.
29.               /* candidate divisible by prime */
30.               /* in list, can't be prime */
31.               is_prime = 0;
32.               break;  /* "for cursor" loop */
33.             }
34.          }
35.          if(is_prime) {
36.
37.            /* add candidate to front of list */
38.            cursor = (INTLIST) malloc(sizeof(INTCELL));
39.            cursor->value = candidate;
40.            cursor->next  = prime_list;
41.            prime_list = cursor;
42.          }
43.         }
44.         return prime_list;
45.        }
```




b. T = {
t1 = {1,2,3,5,6,8,7,6,8,9,10,11,4,3,12}
}

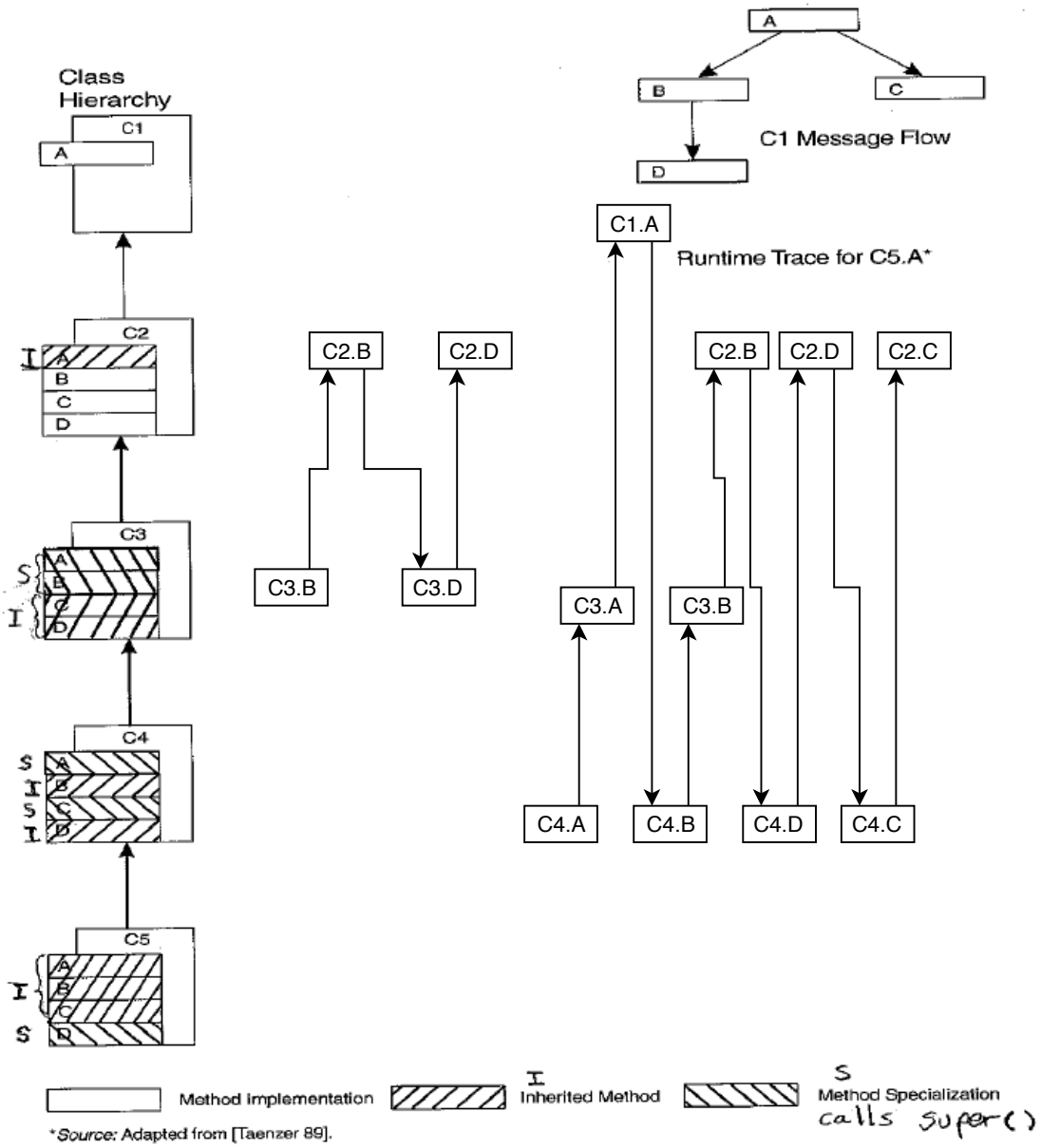
c. T = {
t1 = {1,2,3,5,6,8,7,6,8,9,10,4,3,5,6,8,9,10,11,4,3,12}
}

d. In general, 100% node and edge coverage is not possible. It is possible to have code that looks like the following

```
if(x<2){
   if(x>2){
      print("hi")
   }
}
```

in the above scenario the code will compile however the print statement will never be executed. This will cause at least 1 node and 1 edge to never be hit.

2.a.




b.

When C1.D is called given the above class hierarchy the program will attempt to call C1.D. When it discovers no such call it will move down to C2.D and attempt to call that method. Upon finding the method in C2 it will execute C2.D.

3.a. Pass the function an int of 1 for i.

b. Pass the function an int less than 1 for i.

c. Pass the function any int greater than 1.