Chapter 17

17.1

a) 9

b) $n - 1$

c) 15

d) $2^n - 1$

e) 177

f) number of calls for Fib(n) = (Fib(n − 1) + Fib(n − 2) + 1)

g) No, Ps only mark the path from the start to the exit once a path is found

17.3

The total number of squares in the maze minus one

17.5

a.1) The result is 0.
a.2) The result is 2.
a.3) The result is 0.

b) Power(a, b) = Floor($\log_b$ a)
c)

--
frame pointer
retaddr to Power
1
7
frame pointer
retaddr to Power
0
11
7

17.7

a) The activation record for SevenUp occupies 4 slots (8 bytes). With 16Kbytes allocated to the stack, the largest input value that will work is 2048 (assuming the activation record of main is inconsequential).

b) Again, if the activation record of SevenUp occupies 8 bytes, the a 4KB stack can accommodate SevenUp (512).

17.9

```
/*
** This function returns the position of 'item' if it exists
** between list[start] and list[end], or -1 if it does not.
*/
int BinarySearch(int item, int list[], int start, int end)
{
   int middle = (end + start) / 2;

   /* Did we not find what we are looking for? */
   if (end < start)
      return -1;

   /* Did we find the item */
   else if (list[middle] == item)
      return middle;

   /* Should we search the first half of the array? */
   /* NOTE: The following line is changed from 17.16 */
   else if (item > list[middle])
      return BinarySearch(item, list, start, middle - 1);

   /* Or should we search the second half of the array? */
   else
      return BinarySearch(item, list, middle + 1, end);
}
```

17.11

```
int M()
{
   int num = 1;
   int x = 0;

   while (num > 0) {
       printf("Type a number: ");
       scanf("%d", &num);

       if (num > x)
          x = num;
   }
   return x;
}
```

17.13

Many possible solutions. The recursive solutions will involve recursive depth-first search with backtracking, similar to the maze solution provided in Figure 17.19.