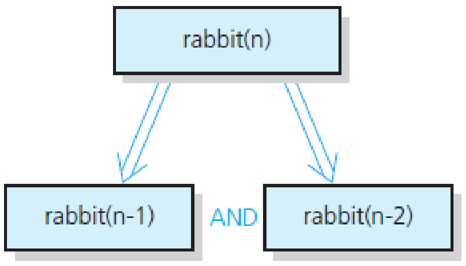
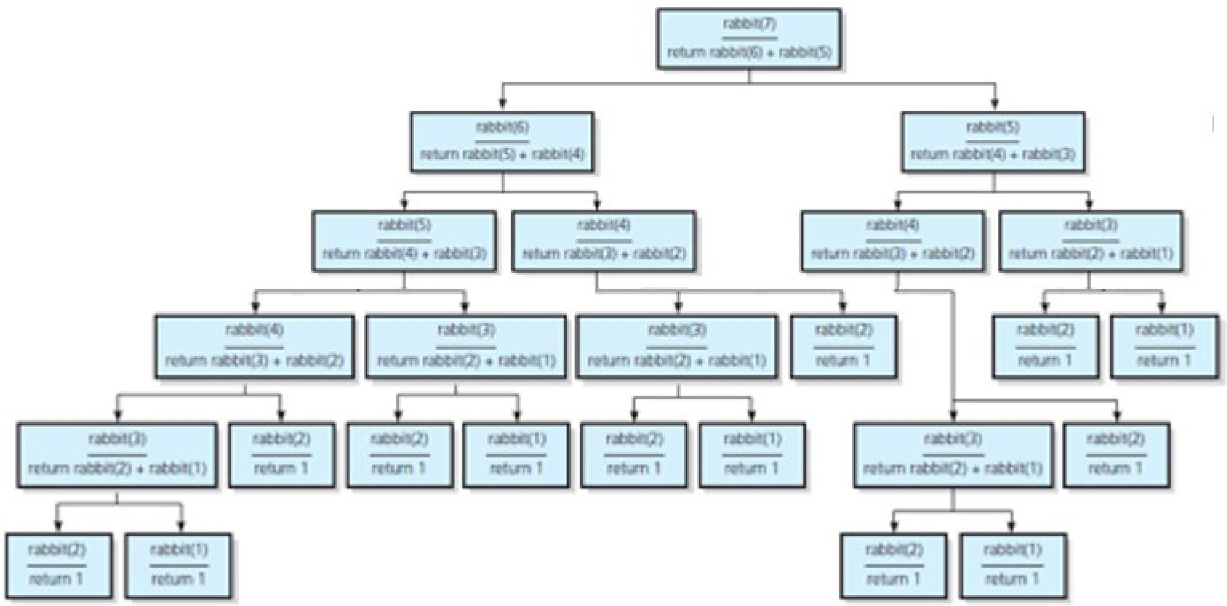
Chapter 2 - Recursion 2

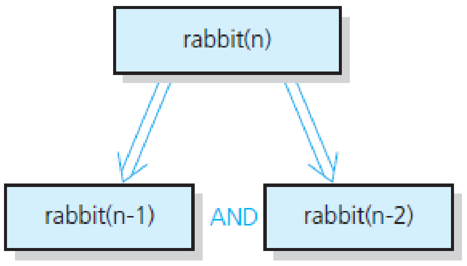
- **Recursive Solutions**
  - A **recursive function** calls by itself.
  - Each recursive call solves an identical, but smaller, problem.
  - Test for base case enables recursive calls to stop.
  - Eventually, one of smaller problems must be the base case.

- The Fibonacci Sequence (Multiplying Rabbits)
  - Assume the following “facts” …
  - Rabbits never die.
  - Rabbit reaches sexual maturity at beginning of third month of life.
  - Rabbits always born in male-female pairs. At beginning of every month, each sexually mature male-female pair gives birth to exactly one male-female pair.

  Monthly sequence
  1. One pair, original two rabbits
  2. One pair still
  3. Two pairs (original pair, two newborns)
  4. Three pairs (original pair, 1 month old, newborns)
  5. Five pairs …
  6. Eight pairs …

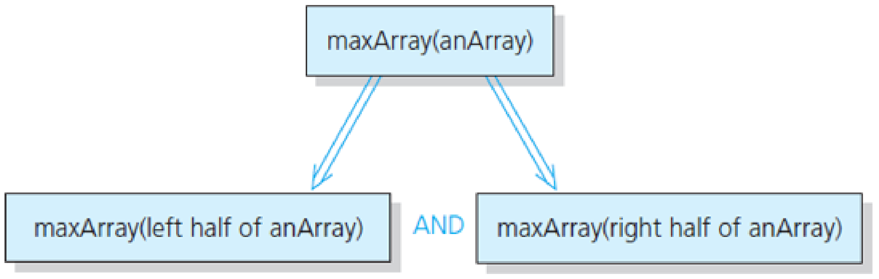


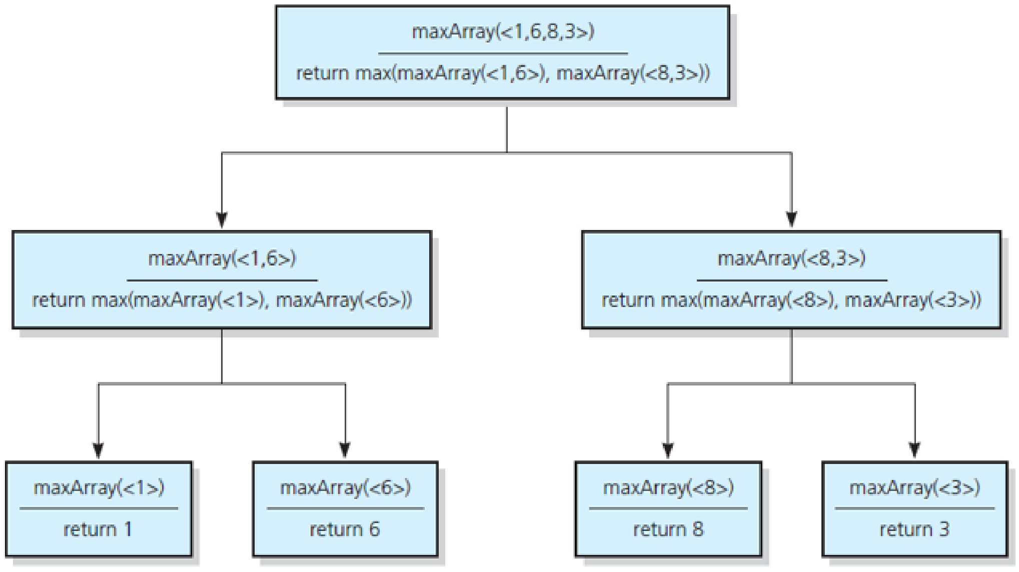

rabbit(7)
return rabbit(6) + rabbit(5)
rabbit(6)
return rabbit(5) + rabbit(4)
rabbit(5)
return rabbit(4) + rabbit(3)
rabbit(5)
return rabbit(4) + rabbit(3)
rabbit(4)
return rabbit(3) + rabbit(2)
rabbit(4)
return rabbit(3) + rabbit(2)
rabbit(3)
return rabbit(2) + rabbit(1)
rabbit(4)
return rabbit(3) + rabbit(2)
rabbit(3)
return rabbit(2) + rabbit(1)
rabbit(3)
return rabbit(2) + rabbit(1)
rabbit(2)
return 1
rabbit(2)
return 1
rabbit(1)
return 1
rabbit(3)
return rabbit(2) + rabbit(1)
rabbit(2)
return 1
rabbit(2)
return 1
rabbit(1)
return 1
rabbit(2)
return 1
rabbit(1)
return 1
rabbit(3)
return rabbit(2) + rabbit(1)
rabbit(2)
return 1
rabbit(2)
return 1
rabbit(1)
return 1
rabbit(2)
return 1
rabbit(1)
return 1



```
//n is a positive integer
int rabbit(int n)
{
    if (n <= 2)
        return 1;
    else // n > 2, so n - 1 >0 and n - 2 > 0
        return rabbit(n - 1) + rabbit(n - 2);
}
```

- **Finding the Largest value in an array**


maxArray(anArray)
maxArray(left half of anArray)
AND
maxArray(right half of anArray)



Find the max in anArray

```
maxArray(anArray)
{
   if(anArray has only one entry)
      return the entry in anArray
   else if(anArray has more than one entry)
      return the maximum of maxArray(left half of anArray) and
                             maxArray(right half of anArray)
}
```

- **Recursion and Efficiency**

Recursion is a powerful problem-solving technique that often produces very clean solutions to even the most complex problems. Recursion solution can be easier to understand and to describe than iterative solutions. By using recursion, you can often write simple, short implementations of your solution.

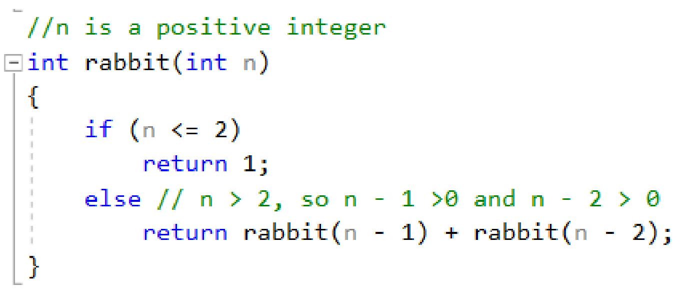
Two factors contribute to the inefficiency of some recursive solutions:

- The overhead associated with function calls
- The inherent inefficiency of making recursive calls whose arguments are the same as those in easier recursive calls

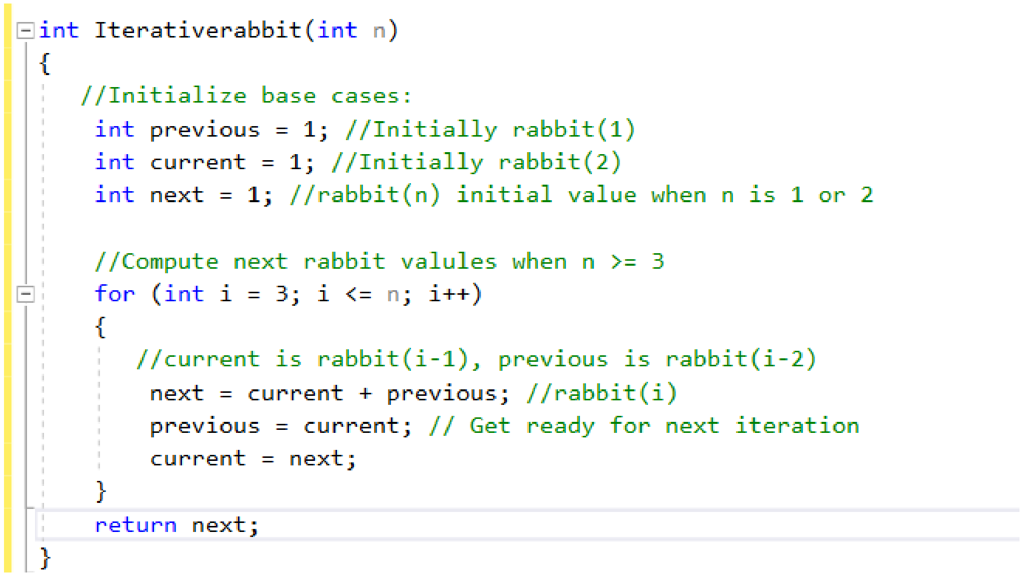
Keep in mind

- Recursion can clarify complex solutions… but…
- Clear, effect iterative solution may be better

Example)



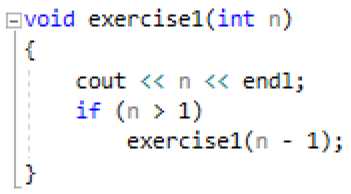
```
//n is a positive integer
int rabbit(int n)
{
    if (n <= 2)
        return 1;
    else // n > 2, so n - 1 >0 and n - 2 > 0
        return rabbit(n - 1) + rabbit(n - 2);
}
```



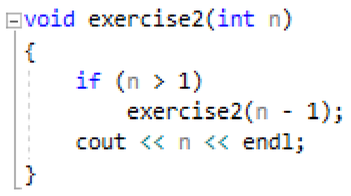
```
int Iterativerabbit(int n)
{
   //Initialize base cases:
    int previous = 1; //Initially rabbit(1)
    int current = 1; //Initially rabbit(2)
    int next = 1; //rabbit(n) initial value when n is 1 or 2

    //Compute next rabbit valules when n >= 3
    for (int i = 3; i <= n; i++)
    {
       //current is rabbit(i-1), previous is rabbit(i-2)
        next = current + previous; //rabbit(i)
        previous = current; // Get ready for next iteration
        current = next;
    }
    return next;
}
```

- **Example**

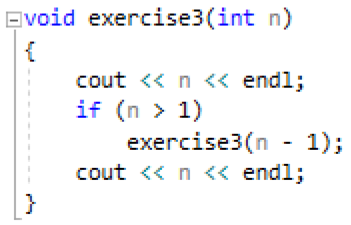
What are the outputs of the following functions with an argument of 3? exercise1 (3), exercise2 (3), and exercise3 (3)?



```
void exercise1(int n)
{
    cout << n << endl;
    if (n > 1)
        exercise1(n - 1);
}
```



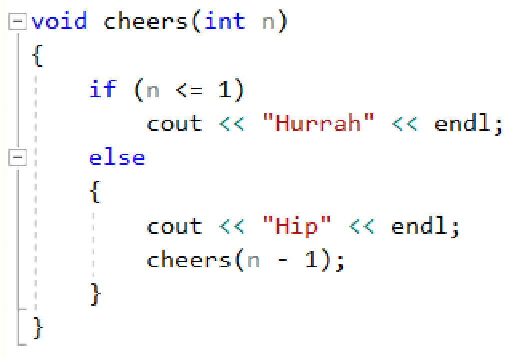
```
void exercise2(int n)
{
    if (n > 1)
        exercise2(n - 1);
    cout << n << endl;
}
```



```
void exercise3(int n)
{
    cout << n << endl;
    if (n > 1)
        exercise3(n - 1);
    cout << n << endl;
}
```

Quiz – 9/3/2021

1. What are the output of the following function with an argument of 3? cheers (3)



```
void cheers(int n)
{
    if (n <= 1)
        cout << "Hurrah" << endl;
    else
    {
        cout << "Hip" << endl;
        cheers(n - 1);
    }
}
```

2. Modify the cheers function from the previous question so that if first prints “Hurrah”