Week 11 Exercise Solution (Question 2 and 3)

Q2.

```
int addBetween (int x, int y) // Precondition: y >= x
{
	if (x > y)
		return 0;
	else
		return x + addBetween(x + 1, y);
}

void printDownToZero (int n)
{
	// base case: n < 0, do nothing

	if (n >= 0)   // recursive case
	{
		cout << n << " ";
		printDownToZero (n-1);
	}
}

void printOddUpTo (int n)
{
	// base case: n < 1, do nothing

	if (n >= 1)   // recursive case
	{
		if (n % 2 == 0)      // n could be even, then we start with n-1
			n--;         // this statement will only execute once.
		printOddUpTo (n-2);
		cout << n << " ";    // output only starts when backtracking starts
	}
}
```

Q3.

```
int binarySearch (int key, int arr[], int first, int last);
// Search arr from index first to last for the key

// Precondition: The array elements arr[first] through arr[last] have values; first <= last;
//                arr is already sorted: arr[first] <= arr[first+1] <= ... <= arr[last].
// Postcondition: if the key is found in arr, its index is returned; otherwise, -1 is returned


int binarySearch (int key, int arr[], int first, int last)
{
	int location = -1;
	if (first <= last)
	{
		int mid = (first + last) / 2;
		if (key == arr[mid])
			location = mid;
		else if (key < arr[mid])
			location = binarySearch(key, arr, first, mid-1);
		else
			location = binarySearch(key, arr, mid+1, last);
	}
	return location;
}
```