CS 162C++
Recursion exercises

Converting a string to an integer

If you read in a string that contains digits, like “342”, then you need to change it to an integer and apply the appropriate multiplying to get 3 * 10^2 + 4 * 10 + 2.

You can do this iteratively with something like this:

```
int converter(string input)
{
    const char ZERO = '0';
    int result = 0;
    while (input.length() > 0)
    {
        char next = input[0];
        result *= 10;
        result += next - ZERO;
        input = input.substr(1);
    }
    return result;
}
```

Write a recursive version of converter.

Reversing a string

Write a recursive program that is given a string as an input and returns the string in reverse order. An iterative version of this is:

```
string reverser(string input)
{
    string result = "";
    int length = static_cast<int>(input.length());
    while (length > 0)
    {
        length--;
        result += input[length];
        input = input.substr(0, length);
    }
    return result;
}
```

Linear Search

Another simple recursive problem is to do a linear search on an array. The iterative version of it looks like this:

```
bool linearFind(int array[], int length, int value)
{
   for (int i = length-1; i >= 0; i--)
   {
      if (array[i] == value)
         return true;
   }
   return false;
}
```

Note that this particular program starts at the last location in the array and searches to zero. This is to make the recursive version simpler to program with the same arguments.

Now write a recursive version of this.

Binary Search

Like with a linear search, a binary search of an array is a simple recursive program to create. Here is an iterative version:

```
bool binFind(int array[], int length, int value)
{
   int low = 0;
   int high = length - 1;
   while (low <= high)
   {
      int middle = (low + high) / 2;

      if (array[middle] == value)
         return true;

      if (array[middle] > value)
         high = middle - 1;
      else
         low = middle + 1;
   }
   return false;
}
```

Write a recursive version of this, note that the recursive version will need an extra parameter, the index of the lowest value to be searched and you want to pass in the index of the highest value instead of length.

Display all Subsets

A classic recursive problem is to display all the subsets of a given set of characters. For example:

```
AB => AB, A, B, {}

ABC => ABC, AB, AC, A, BC, B, C, {}

ABCD => ABCD, ABC, ABD, ACD, AB, AC, AD, A, BCD, BC, BD, B, CD, C, D, {}
```

For this exercise, you should input a string and output all the substrings of it. You do not need to display the empty string.

Display all permutations

Another classic problem is to display all the permutations of a given set of characters. For example:

```
AB => AB, BA

ABC => ABC, ACB, BAC, BCA, CAB, CBA

ABCD => ABCD, ABDC, ACBD, ACDB, ADBC, ADCB, BACD, BADC, BCAD, BCDA, BDAC, BDCA
        CABD, CADB, CBAD, CBDA, CDAB, CDBA, DABC, DACB, DBAC, DBCA, DCAB, DCBA
```

Write a recursive program that performs this action on an input string.