1. Number representation

   a) 1999
      Add the one bits: \(= 1024 + 512 + 256 + 128 + 64 + 8 + 4 + 2 + 1\)
      or subtract the zero bits from the word of all 1's: \(= 2047 - 32 - 16\)

   b) 11111001111 = 111 1100 1111 = 7CF

   c) 11111011011 (for 2011)

2. Short Answer

   a) Set to zero, or clear the memory location.

   b) 90.
      Java methods use call by value for primitive types such as \texttt{int}. Once the method is
      finished with execution the value of \texttt{grade} will be the same as before the method was
      called.

   c) i. This is an ill-conditioned problem.

3. Input and Output

   a) TACGTACG

   b) The most common solution was as follows:

   ```java
   public class Complement {
       public static void main ( String[] args ) {
           while(StdIn.isEmpty() == false) {
               char base = StdIn.readChar();
               if(base == 'A')
                   System.out.print('T');
               else if(base == 'T')
                   System.out.print('A');
               else if(base == 'G')
                   System.out.print('C');
               else if(base == 'C')
                   System.out.print('G');
           }
           System.out.println();
       }
   }
   ```

   c) java Complement < sequence.txt
4. Arrays and Loops

```java
public class Inventory {
    public static void main(String[] args) {

        // read the number of records
        int N = StdIn.readInt();

        // initialize four parallel arrays
        String[] make = new String[N];
        String[] model = new String[N];
        double[] cost = new double[N];
        double[] sale = new double[N];

        double profit;
        int maxIndex = 0; // index of item with max profit
        double maxProfit = -1.0; // max profit

        // read in the data
        for (int i = 0; i < N; i++) {
            make[i] = StdIn.readString();
            model[i] = StdIn.readString();
            cost[i] = StdIn.readDouble();
            sale[i] = StdIn.readDouble();

            // calculate the profit
            profit = sale[i] - cost[i];

            // save index of record with max profit
            if (profit > maxProfit) {
                maxIndex = i;
                maxProfit = profit;
            }
        }

        // print the record of the item with the maximum profit
        System.out.println(''Make: '' + make[maxIndex] +
            '' , Model: '' + model[maxIndex] +
            '' , Cost: '' + cost[maxIndex] +
            '' , Sale: '' + sale[maxIndex] +
            '' , Profit: '' + maxProfit);}
    }
}
```
5. Recursion

a) The program will print \texttt{a[0]} on line 4, then return from the \texttt{mystery} method on line 7 because \texttt{x < 2}, and finally print \texttt{a[0]} again on line 26:

\begin{verbatim}
The
The
\end{verbatim}

b)
\begin{verbatim}
mystery (a, 5) call on line 25
  mystery (a, 2) call on line 12
    mystery (a, 1) call on line 15
      return 1 return on line 7
    return 2 return on line 15 (1 + 1)
  return 1 return on line 12 (2 - 1)
\end{verbatim}

c)
\begin{verbatim}
and
is
truth
truth
\end{verbatim}

6. Sorting and Analysis

a(i) \( N \).
The first iteration of the while loop will go through the for loop \( N \) times, and then exit the while loop because swap is still false.

a(ii) \( N^2 \).
The while loop will iterate \( N \) times, and the for loop will iterate \( N \) times for each iteration of the while loop.

b(i) \( N \).
The first call to \texttt{isSorted()} will iterate through the for loop \( N \) times, return true, and exit the while loop.

b(ii) \( N! \).
The number of permutations of the array (\( N! \)) determines how many times the while loop will have to run.
7. Recursive Graphics
8. Java Programming, Recursion, Functions, and Loops

```java
public static int gcd(int p, int q) {
    while (q != 0) {
        int mod = p % q;
        p = q;
        q = mod;
    }
    return p;
}
```

9. Definition

An algorithm is a precisely-specified procedure for solving a problem.