In the decimal number system, the position of a digit is used to signify the power of 10 that digit is to be multiplied with. For example, "314" denotes the number $3 \times 100 + 1 \times 10 + 4 \times 1$. The base b number system generalizes the decimal number system: the string "$a_{k-1}a_{k-2}\ldots a_1a_0$", where $0 \le a_i < b$, denotes in base-b the integer $a_0 \times b^0 + a_1 \times b^1 + a_2 \times b^2 + \cdots + a_{k-1} \times b^{k-1}$.

Write a program that performs base conversion. The input is a string, an integer b_1, and another integer b_2. The string represents be an integer in base b_1. The output should be the string representing the integer in base b_2. Assume $2 \le b_1, b_2 \le 16$. Use "A" to represent 10, "B" for 11, ..., and "F" for 15. (For example, if the string is "615", b_1 is 7 and b_2 is 13, then the result should be "1A7", since $6\times7^2+1\times7+5 = 1\times13^2+10\times13+7$.)

Hint: What base can you easily convert to and from?

Solution: A brute-force approach might be to convert the input to a unary representation, and then group the 1s as multiples of b_2, b_2^2, b_2^3, etc. For example, $(102)_3 = (1111111111)_1$. To convert to base 4, there are two groups of 4 and with three 1s remaining, so the result is $(23)_4$. This approach is hard to implement, and has terrible time and space complexity.

The insight to a good algorithm is the fact that all languages have an integer type, which supports arithmetical operations like multiply, add, divide, modulus, etc. These operations make the conversion much easier. Specifically, we can convert a string in base b_1 to integer type using a sequence of multiply and adds. Then we convert that integer type to a string in base b_2 using a sequence of modulus and division operations. For example, for the string is "615", $b_1 = 7$ and $b_2 = 13$, then the integer value, expressed in decimal, is 306. The least significant digit of the result is 306 mod 13 = 7, and we continue with 306/13 = 23. The next digit is 23 mod 13 = 10, which we denote by 'A'. We continue with 23/13 = 1. Since 1 mod 13 = 1 and 1/13 = 0, the final digit is 1, and the overall result is "1A7".

```
string ConvertBase(const string& s, int b1, int b2) {
  bool is_negative = s.front() == '-';
  int x = 0;
  for (size_t i = (is_negative == true ? 1 : 0); i < s.size(); ++i) {
    x *= b1;
    x += isdigit(s[i]) ? s[i] - '0' : s[i] - 'A' + 10;
  }

  string result;
  do {
    int remainder = x % b2;
    result.push_back(remainder >= 10 ? 'A' + remainder - 10 : '0' + remainder);
    x /= b2;
  } while (x);
```

```
  if (is_negative) {  // s is a negative number.
    result.push_back('-');
  }
  reverse(result.begin(), result.end());
  return result;
}
```

The time complexity is $O(n(1 + \log_{b_2} b_1))$, where n is the length of s. The reasoning is as follows. First, we perform n multiply-and-adds to get x from s. Then we perform $\log_{b_2} x$ multiply and adds to get the result. The value x is upper-bounded by b_1^n, and $\log_{b_2}(b_1^n) = n \log_{b_2} b_1$.

7.3 Compute the spreadsheet column encoding

Spreadsheets often use an alphabetical encoding of the successive columns. Specifically, columns are identified by "A", "B", "C", ..., "X", "Y", "Z", "AA", "AB", ..., "ZZ", "AAA", "AAB", ...

Implement a function that converts a spreadsheet column id to the corresponding integer, with "A" corresponding to 1. For example, you should return 4 for "D", 27 for "AA", 702 for "ZZ", etc. How would you test your code?

Hint: There are 26 characters in ["A", "Z"], and each can be mapped to an integer.

Solution: A brute-force approach could be to enumerate the column ids, stopping when the id equals the input. The logic for getting the successor of "Z", "AZ", etc. is slightly involved. The bigger issue is the time-complexity—it takes 26^6 steps to get to "ZZZZZZ". In general, the time complexity is $O(26^n)$, where n is the length of the string.

We can do better by taking larger jumps. Specifically, this problem is basically the problem of converting a string representing a base-26 number to the corresponding integer, except that "A" corresponds to 1 not 0. We can use the string to integer conversion approach given in Solution 7.1 on Page 91

For example to convert "ZZ", we initialize result to 0. We add 26, multiply by 26, then add 26 again, i.e., the id is $26^2 + 26 = 702$.

Good test cases are around boundaries, e.g., "A", "B", "Y", "Z", "AA", "AB", "ZY", "ZZ", and some random strings, e.g., "M", "BZ", "CCC".

```
int SSDecodeColID(const string& col) {
  int ret = 0;
  for (char c : col) {
    ret = ret * 26 + c - 'A' + 1;
  }
  return ret;
}
```