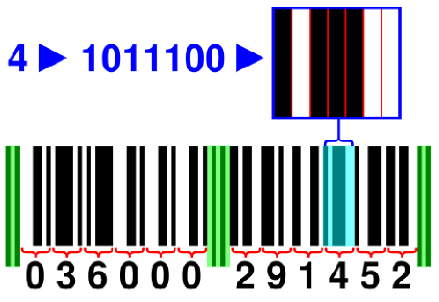
Computer Science 3-4
Problem Set #12: Barcodes!

First off you are going to do a quick little activity on Barcodes, before you delve into this homework assignment…actually do it, if you skip it you will ask me silly questions like “how do barcodes work” and I will just tell you to actually read the assignment! So take a look at this:

Make your own Barcodes!

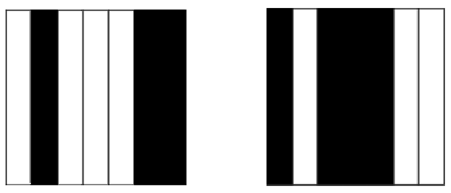
We see barcodes almost everywhere we look these days. Their purpose is simple, to provide an easy way to scan in a string of numbers, without having to bother to type them in. There are many varieties of barcodes (it is really just a general term to describe a black and white pattern that represents data), but the most common one that we see is the Universal Product Code (UPC)…this is the code that is on almost everything we buy in the super market. In this worksheet we’ll use the codes for the 12 digit UPC, commonly called the UPC-A.



Each number is actually represented by a grouping of 7 bars. They touch, so 3 bars together looks like one fat bar. The shaded area on the ends and middle are called “guard bars,” they help the optical scanner detect the actual numbers. There are two sets of codes: one for the left hand numbers (the first 6 digits) and one for the right hand numbers (the last 6 digits). This is done to make sure there a clear distinction between the company and product identification codes. The table below indicates how the patterns are drawn (0 means leave the bar white, 1 means color the bar black)

Left Hand Codes		**Right Hand Codes**	
0	0001101	0	1110010
1	0011001	1	1100110
2	0010011	2	1101100
3	0111101	3	1000010
4	0100011	4	1011100
5	0110001	5	1001110
6	0101111	6	1010000
7	0111011	7	1000100
8	0110111	8	1001000
9	0001011	9	1110100

Example: Writing “4” using the left and right hand codes



Use the table above to fill out a barcode for the number: 056321 079805.

M

Now suppose we want to make our own UPC-A barcodes, we need to know a little bit about
together.

> Assume the 12 digits are d_1, d_2, d_3, ..., d_{11}, d_{12}. In a valid UPC barcode, d_{12} is so selecte
> following identity true:
>
> $$(3\cdot d_1 + d_2 + 3\cdot d_3 + d_4 + ... + 3\cdot d_{11} + d_{12}) \bmod 10 = 0.$$

Another way of saying this is that: "the sum of the even-position digits, plus three times the
position digits equals a multiple of 10."

This is a quick way for the computer to check for fake barcodes or to make sure that it corre
numbers it scanned….when the cashier has to rescan the item it usually means this check fai

In the example above, we had the UPC code 056321 079805, let's see if it checks out!

3(0) + 5 + 3(6) + 3 + 3(2) + 1 + 3(0) + 7 + 3(9) + 8 + 3(0) + 5 = 80

Looks good!

For the next two problems find the missing digit:

1) 5 7 3 3 1 0 0 _ 3 1 1 9

2) 2 0 0 1 2 1 9 3 _ 4 5 7

Finally make your own proper UPC-A codes:

Code 1

M

Code 2

M

The first few problems will involve the good old fashioned console…we'll get to the graphics a little bit later.

Problem 1: Write a program that will ask a user to type in the first 11 digits of a bar code and output the proper checksum value for the 12th digit. Do not make 11 variables! Use a vector or array.

Problem 2: Write a program that asks the user to enter in *any* 11 digits of a bar code and then tells the user the value of the missing digit. You may ask the user which place is missing (or they can enter -1 to indicate the "blank" space, whatever your preference). Make sure to test this carefully with several barcodes and several missing places!

Problem 3 (Challenge) – File I/O: For this challenge you will have to teach yourself something useful! Using file input/output is a hugely useful feature of C++ that will allow you to save data to the disk for later use (in this case as a simple .txt file). The first part of this challenge is to find a tutorial on File I/O for C++ (hint: there are quite a few options…but don't use anything that requires an extra download). Once you have this info and using your programs from above, write a new program that will:

a) The user selects if they want to find the 12th digit or any digit.

b) They specify the name of a .txt file to load, which will contain one barcode per line.

c) The program reads the file, finds the missing digits and generates a new file called "results.txt" that has the completed barcodes entered one per line.

Time to draw!

Now we are going to get on to drawing our barcodes to the screen. Before you go on to this section you need to check out this video where I go over how to create a function that lets us draw to the RenderWindow: http://youtu.be/6cAw7AdOPg4

Problem 4: Write a program that will convert a single number into its graphical barcode counterpart. The user will enter if it is on the left or right-hand side and then the digit whose bar pattern you want. Remember that all barcode digits are a pattern of 7 bars (white or black). I would suggest doing this in two parts: First take a minute to find pattern/patterns in how the bars are generated (hint: Left vs. Right is a good place to start)…the more general your description of the bars the less you'll have to hardcode.

Second, make sure you build a function that lets you specify where the bars are drawn…you'll be making a whole 12 digit barcode in the next part! Third, if you haven't taken a look at the switch() statement yet…now's the time!

Problem 5: Take your code from Problem 4 and use it to make a program that will draw an entire bar code that the user types in. The user should be able to enter the code as one number. A couple of tips: remember that the first 6 digits will use the left hand code and the second 6 will use the right hand code and use your number splitter function from a while back to help you out. This task has a lot of moving parts…a strong plan/pseudocode will help you immensely!

Problem 6 (Challenge) – Bzzzt!: Add a feature to your version of problem 5 that checks to see if the barcode is actually valid. If the barcode passes muster, flash a message to the screen that lets the user know everything is great…otherwise put a big red "X" over everything and say something snarky.