Developing programs incrementally

As programs become more complex, the process by which a programmer develops the program becomes more important. What a programmer does not want to do (and yet what many new programmers do) is write the entire program, compile it, and then run it -- hoping the program works. The problem with that process is that if the program does not work (as is almost always the case), debugging can be extra difficult because there may be many distinct bugs.

The object of this lab is to write a program that will both (1) convert of 4-letter "word" into a phone number and (2) convert a 4-digit number into a word, using the keypad arrangement shown at right.

Figure: Letters on keypad.

For example given WART, you'd get the phone number 9278, though 9278 could be any choice of letters from the columns below:

```
WAPT
XBQU
YCRV
Z S
```

So, YBST and XAQU are words with the same number.

This will involve some careful consideration before diving into writing code.

Experienced programmers develop programs incrementally, starting with a simple version of the program, and then growing the program little-by-little into a complete version. Here we explore an example of this.

The following pseudocode for converting a 2-digit number to a word was developed in lab:

1. Get the digits in the number into separate variables (dig1 and dig2, say).
2. Assign appropriate values to the number of possible letters that could be substituted for the digit. In the case of 0 or 1, this is 1; for 2,3,4,5,6,8, it's 3 and for 7 and 9, it's 4.
3. Use conditionals, looping structures and ASCII codes to work out the logic of the conversion. Nested for loops will do nicely and the ASCII code for 'A' is 65, etc.

The (buggy) code we developed in class is on the next page. Here is some output from the code:

```
Convert from number to word or word to number? (n/w): n

Enter a two-digit number: 67

You entered 67
MS
NS
OS
Press any key to continue . . .
```

As you can see, the output that should be MP, MW, MR, MS, NP, NQ, NR, NS, OP, OQ, OR, OS is missing a lot. Your job is to understand how the code works so you can debug it and produce the right output Then scale up to four-digit numbers.

```
// G. Hagopian
// Phone numbers

#include <iostream>

using namespace std;

int main()
{
	char choice, first, second;
	int dig1, dig2, numPoss1, numPoss2, number;
	cout << "\nConvert from number to word or word to number? (n/w): ";
	cin >> choice;
	if(choice == 'n')
	{
		cout << "\nEnter a two-digit number: ";
		cin >> number;
		dig1 = number/10;
		dig2 = number - 10*dig1;
		cout << "\nYou entered " << dig1 << dig2 << endl;
		if(dig1 == 1) numPoss1 = 1;
		else if(dig1 == 7 || dig1 == 9) numPoss1 = 4;
		else numPoss1 = 3;
		if(dig2 == 1 || dig2 == 0 ) numPoss2 = 1;
		else if(dig2 == 7 || dig2 == 9) numPoss2 = 4;
		else numPoss2 = 3;
		for(int i = 0; i < numPoss1; ++i)
		{
			if(dig1 == 1) first = '1';
			else if(dig1 < 7) first = (char)((dig1-2)*3 + 65 + i);
			else if(dig1 == 7) first = (char)(80+i);
			else if(dig1 == 8) first = (char)(84+i);
			else if(dig1 == 9) first = (char)(87+i);
			for(int j = 0; j < numPoss2; ++j)
			{
				if(dig1 == 0) second = '0';
				else if(dig2 == 1) second = '1';
				else if(dig2 < 7) second = (char)((dig1-2)*3 + 65 + j);
				else if(dig2 == 7) second = (char)(80+j);
				else if(dig2 == 8) second = (char)(84+j);
				else if(dig2 == 9) second = (char)(87+j);
			}
			cout << first << second << endl;
		}


	}
	else if(choice == 'w')
	{ // Write code here
	}

}
```