Pyramid of Letters

The Pyramid is a structure found in many cultures. This shape is often associated with supernatural power. You are asked to write a program that accepts a single character from "A" through "Z" and produces an output in the shape of a pyramid composed of the letters up to and including the letter that was input. The top letter in the pyramid should be an "A" and on each level, the next letter in the alphabet should fall between the letter that was introduced in the level above it.

EXAMPLE: (bolded values denote user input)

Please enter a letter (A-Z): **E**

Your pyramid is as follows:

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
```

Are there more letters (Y/N)? N

PrimInt

Given this class:

```
class PrimInt {
      private:
            int i_;
      public:
            PrimInt() : i_(0) { }
            PrimInt(int n) : i_(n) { }
            int value() const; // returns value of i_
            bool zero() const; // returns true if i_ equals zero
            void inc(); // increases i_ by 1
            void dec(); // decreases i_ by 1
            friend istream & operator>> (istream & is, PrimInt & n);
};

int PrimInt::value() const
{
      return i_;
}

bool PrimInt::zero() const
{
      return (i_==0);
}

void PrimInt::inc()
{
      i_++;
}

void PrimInt::dec()
{
      i_--;
}

istream & operator>> (istream & is, PrimInt & n)
{
      is >> n.i_;
      return is; // enables "stacked" input
}
```

Given these rules:
1. Use only the PrimInt class and no other type except char.
   Type char must be used to accept the desired operation and
   to prompt for continuation or termination.
2. No loops allowed, except the loop that repeats if the user chooses to continue.
3. Do not modify the PrimtInt class.

Write a C++ program that will:
1. prompt the user for the desired operation (either add or multiply)
2. reject invalid operators
3. prompt the user for the two values to be operated on
4. perform the operation and output the original values and the answer
5. prompt the user for continuation or termination.

It is okay to place your PrimInt class in the same file as
your main function rather than create separate header and
definition files for the class.

```
SAMPLE RUN:
PrimInt Class Driver
Which operation (+ or *)? #
Invalid operator.
Do you want to continue? (Y/N) y
Which operation (+ or *)? +
Enter the two values to be added: 17 35
The sum of 17 and 35 is 52.
Do you want to continue? (Y/N) y
Which operation (+ or *)? *
Enter the values to be multiplied: 17 35
The product of 17 and 35 is 595.
Do you want to continue? (Y/N) n
Press any key to continue . . .
```

Roman Numeral Calculator

The Romans were noted for many significant advances in math, the sciences, and fine arts. The numbering system they developed used Roman Numerals to represent common numeric values. The values of Roman digits are as follows:

I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Here are the rules for forming Roman Numerals:

1) Digits are combined left to right, larger to smaller.
2) Digits for powers of 10 (I, X, C, and M) may be written 1, 2, or 3 times in succession. Other digits may appear only once in a Roman Numeral.
3) Numerals are evaluated by adding the values of their digits.
4) Certain exceptions allow a smaller to precede a larger, which means subtract the smaller from the larger:
   a) I may precede V which means 4 (5 - 1).
   b) I may precede X which means 9 (10 - 1).
   c) X may precede L which means 40 (50 - 10).
   d) X may precede C which means 90 (100 - 10).
   e) C may precede D which means 400 (500 - 100).
   f) C may precede M which means 900 (1000 - 100).
5) Thus, the Roman Numerals for 1 to 9 are I, II, III, IV, V, VI, VII, VIII, and IX.
   a) Roman Numerals for 10 to 90 (counting by 10) are the same except use X, L, and C.
   b) Roman Numerals for 100 to 900 (counting by 100) are the same except use C, D, and M.
   c) Roman Numerals for 1000 to 3000 (counting by 1000) are M, MM, and MMM.
6) The largest Roman Numeral using this system is MMMCMXCIX, which equals 3999.
7) There are no Roman Numerals for zero or negative integers.

Notice that rules 5, 6, and 7 do not provide any new information. They simply clarify rules 1, 2, 3, and 4.

One variation of the rules for forming Roman Numerals is to allow 4 I's, X's, C's, or M's in succession, thus eliminating the need for the exceptions that require subtraction (IV, IX, etc.). Sorry, but this variation is not allowed in this problem.

Write a C++ program that will allows the user to perform calculations on Roman Numerals. The arithmetic operators that your program should incorporate include +, -, *, and /. For division, assume integer division. Invalid input should be rejected. If the result of an operation would be negative or zero, output "No Solution using Roman Numerals."

Bowling for Fun

Write a program which scores a bowling match. The program should be interactive. The program should request scores frame by frame and produce a final score. Note that for each frame a bowler rolls two balls unless a strike is thrown. Acceptable input is any integer from 0 - 9 and a "/" for a spare (all ten pins on the second ball) and a "X" for a strike (all ten pins on the first ball).

Error checking should guard against improper inputs, including the fact that, at most, ten pins can be knocked down in a frame.

For example:

```
First ball: 7
Second ball: 4
Not possible only 3 pins left
Second ball: 3  (an equivalent input here is / )
```

Scoring Rules:

- There are 10 frames in a game.
- Score for a given frame is the sum of pins on both balls unless a spare or strike is scored.
- If a spare is scored, the score for that frame is 10 plus the number of pins on the first ball of the next frame.
- If a strike is scored, the score for that frame is 10 plus the number of pins on the next two balls.
- A spare scored in the tenth frame earns the player one extra ball.
- A strike scored in the tenth frame earns the player two extra balls.

Once all frames have been entered, your program should print a listing consisting of five lines: (i) frame number, (ii), first ball score, (iii) second ball score, (iv) frame score, and (v) running score.

Example:

Frame #	1	2	3	4	5	6	7	8	9	10	Xtra1	Xtra2
First Ball	6	5	7	8	X	X	9	6	7	X	X	X
Second Ball	2	3	/	/			/	1	2			
Frame Score	8	8	18	20	29	20	16	7	9	30		
Running Score	8	16	34	54	83	103	119	126	135	165		