1. Write a C++ program that prints a table of the characters for the ASCII codes from \( ' ' \) (space) to \( '~' \) (tilde). Print the characters 16 to a line separated by spaces. Do not use the ASCII codes (e.g., space is 32, ~ is 126) to control the loop. Instead, use the characters themselves (e.g., ' ' or '~') to control the loop. (This is generally going to be the right way to do something with ASCII characters – use the characters not the codes!) **Do not print a trailing space on any line (except perhaps the last one after the '~').** This illustrates the ‘fence post’ problem – you need 15 spaces and 1 end-of-line on each line. (20 points)

The output should look like this:

```
! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

Notice I did not print a space after the last ~. You may, but you may not have a space after any other terminal character on a line. This is ONE loop. To determine whether or not to print a space, COUNT the number of things you print on the line. Do not check for '/', '?', etc. to determine the end of the lines.

2. A worker is paid one cent for the first day’s work, 2 cents for the second, 4 cents for the third, and so on with, the integer pay in pennies being doubled each day. At the end of the job, she gets a single lump sum paycheck in dollars and cents (this is a floating-point value which needs a lot of digits of precision). Accumulate the pay in an int, and only convert to a floating-point type to print the total pay. Write a program that prompts the user for a number of days, then prints each day’s pay in pennies, followed by the total accrued pay in dollars with a fractional part for cents at the end. Round the displayed pay to two decimal places. Test with several different numbers of days, including 0 and 64. Work out (please – do not just try it until it fails, solve this first!) at how many days the program stops working correctly with a 32-bit int for the number of pennies, and include a test for that number of days and a test for a larger number of days. (20 points)

A test run should look something like this for 8 days:

```
How many days? 8
Pay for day 1 = 1
Pay for day 2 = 2
Pay for day 3 = 4
Pay for day 4 = 8
Pay for day 5 = 16
Pay for day 6 = 32
Pay for day 7 = 64
Pay for day 8 = 128
Total pay is $2.55
```