

CSE 130 Fall 2009

Programming Assignment 2

Due Date: 11am, Thursday Oct 8, 2009. Use `~cse130/submit` command from your sparky account. Do not forget to change to your working directory first. Include your name and Id number as a comment. Make sure that your program compiles with no errors and runs on sparky.

This assignment builds on whatever we did for program-1. There are two parts to this assignment, but it is a single program as before. **Read input only once at the beginning.** If you are not able to do both parts, you may submit your program with only one part that is working and get partial credit. Make sure that you declare all variables, and initialize them properly. Write plenty of comments in your program. Some points are reserved for that.

First prompt the user asking for two positive (> 0) integers m and n . Use a `printf` statement for prompt. Read these integers using `scanf` function. Assume that user inputs correct values. **THERE IS NO NEED FOR ERROR CHECKING.**

After read input calculate product $k = m*n$. Next choose smaller of m and n , and call it `small`. Call the other value `large`. Print values `small` and `large` as shown in the sample output. For example, if $m = 4$ and $n = 11$ then `small = 4`, and `large = 11`. Note that for part-2 we need `small` to be less than or equal to 9. But we do not check that.

Part 1: For the first part, sum all integers between `large` and product k (both included) that are NOT divisible by 3. Write a for-loop that runs from $i = large$ to $i = product$. Inside the loop, use an if statement with condition $(i\%3 \neq 0)$ for selecting integers for summation and count-update. Count how many such integers are there. **You MUST use a for loop.** When you get out of loop, print sum and count using `printf` function. For example, if $m = 4$ and $n = 11$, we have $k = 44$. Integers not DIVISIBLE by 3 (between 11 and 44) are: 11, 13, 14, 16, 17, 19, 20, ..., 41, 43, 44. Their sum is 638, and the count is 23. For some values of m and n , it is possible that there will be no such integers between m and n that is divisible by k . So both sum and count would be zero.

Part 2: For part 2, write nested loops to print a parallelogram like figure using integer `small`. This figure is made up of spaces and single digit integers. Remember `small` has value 4.

Parallelogram for size 4 is printed below.

```
  1 1 1 1
 2 2 2 2
 3 3 3 3
4 4 4 4
```

Note here that some leading blanks or spaces (" ") are printed first, followed by a single digit integer repeated several times. Since `small = 4`, this figure has exactly 4 lines. On the first line we have $2*(small - 1) = 6$ leading spaces. On the second line it is $2*(small - 2) = 4$ and so on. On the 4th line no leading spaces are printed. So leading spaces decrease 6, 4, 2, and 0. If m were 8, we would have 14, 12, 10, 8, ..., 0 leading spaces for 8 lines respectively

Printing of single digit integer is again decided by `small`. On the first line, we print 1 followed by a single space/blank (" "), repeated 4 times (since `small = 4`). For this use **format specifier or control string "%1d "**. Here `1d` means room for only 1 digit. After this there is a single blank in the `printf`

control string. On the next line, print integer 2 followed by a single space repeatedly. Since small = 4, on each line we have single digit integers repeated 4 times. Also, this is done repeatedly for 4 lines.

You will need an outer loop for printing each line. This loop should print number of lines = value in small. Declare some variable i as a line counter and use it to control the outer loop. Inside this you will have two inner loops, the first one is for printing leading spaces. The second inner loop prints single digit integers **using "%1d "** format, repeatedly. Again you must control how many spaces are printed on each line followed by correct number of integers. The first inner loop prints 2*(small - 1) leading spaces on the first line, and no spaces (zero) on the last line. Use line counter to reduce number of spaces on successive lines. Use some counter, say j, to control number of integers printed on each successive lines. For inner and outer loops, you may use a for loop or a while loop or their combination.

Note that if small = 1, figure would have only one line of with no leading spaces. But it will have 1 followed by a blank. See class notes for examples of nested loops that print stars.

Sample Session (on sparky):

Type in integers m and n.

7 3

small = 3, and large = 7

Sum = 135 and Count = 10

Parallelogram for size 3 is printed below.

```
  1 1 1
 2 2 2
3 3 3
```

Program structure is on the next page.

Program Structure:

```
// Program-2, date
// Your name id etc.

#include <stdio.h>
int main (void)

{
    // Variable declarations for both parts.

    printf("Type in ..... .. ");
    scanf("%d%d", &m, &n);

    // Calculate product k, small and large.
    // Print small and large.

    // Part-1:
    Calculate Sum and Count using a loop.
    Use an if statement to select numbers for
    summation and updating count.

    // Print Sum and Count.

    // Part-2:

    // Nested loops for printing the parallelogram.
    // Outer loop to control number of lines.
    // { Inner-loop-1 for printing leading spaces. }
    // { Inner-loop-2 for printing single-digit integers. }
    // Print \n.
    // Outer-loop ends. Program ends.

    return 0;
}
```