PDS Lab 2020-21 Spring Section 4

Week 2: 7-April-2021 (Scan, print, arithmetic opr.)

w02-1. Write a C program that takes as input two integer elements from the keyboard, computes and a/b, and prints their respective values on the terminal.

- The value of a/b should be correct up to 6th decimal place.
- The name of your C file should be w02-1.c.

Example

```
Enter a and b: 2 3
a+b = 5
a-b = -1
a*b = 6
a/b = 0.666667
```

w02-2. Write a program that takes as input the value of a floating-point variable, say x, calculates the value of the expression $(1 + x + x^2 + x^3 + ... + x^7)$, and prints the result on the terminal correct up to 10^{th} decimal place.

- Assume that the value of x is positive and less than 1.
- You cannot use math library.
- The name of your C file should be w02-2.c.

Examples

```
Enter x: .5
Answer = 1.9921875000

Enter x: .9
Answer = 5.6953272820

Enter x: .9999
Answer = 7.9970202446

Enter x: .999999
Answer = 8.0000000000
```

w02-3. User supplies a positive integer n having value at most 9999. Your program should shift its digits cyclically towards right, one digit at a time, and should print the result on the terminal. This should continue until you get back the original value of n.

Examples

```
Enter n: 2345
5234
4523
3452
2345
```

```
Enter n: 3251
1325
5132
2513
3251
```

```
Enter n: 23
3002
2300
230
23
```

Week 3: 21-April-2021 (if-else, switch, loop)

w03-1. Given the day, month, year as integers, print the season. You should use **`switch-case`**.

- Winter: December to February.
- Summer: March to May.
- Monsoon: June to September.
- Autumn: October to November.

Examples

```
Enter the day, month, year: 14 4 2021
It's Summer!

Enter the day, month, year: 13 1 2022
It's Winter!

Enter the day, month, year: 15 5 2022
It's Summer!

Enter the day, month, year: 19 9 2021
It's Monsoon!

Enter the day, month, year: 1 10 2022
It's Autumn!
```

w03-2. Given a cubic polynomial $f(x) = a + bx + cx^2 + dx^3$ and an interval $[u, v]$, check whether there exists any root in the interval $[u, v]$. If it exists, then find whether there are any in $[u, \frac{1}{2}(u+v)]$ and in $[\frac{1}{2}(u+v), v]$.

- Consider all numbers in floating point.
- Use the fact that $f(x) = 0$ for some $x \in [u, v]$ if one of $f(u)$ and $f(v)$ is not negative and the other is not positive.

Examples

```
Enter a, b, c, d: 1 -1 2 -3
Enter u, v: -1 1
[-1.000000, 1.000000]: YES
[-1.000000, 0.000000]: NO
[0.000000, 1.000000]: YES

Enter a, b, c, d: 1 -1 2 -3
Enter u, v: 0 1
[0.000000, 1.000000]: YES
[0.000000, 0.500000]: NO
[0.500000, 1.000000]: YES

Enter a, b, c, d: 1 -1 2 -3
Enter u, v: 0.5 1
[0.500000, 1.000000]: YES
[0.500000, 0.750000]: NO
[0.750000, 1.000000]: YES
```

```
Enter a, b, c, d: 13 -17 19 0
Enter u, v: -1000 1000
[-1000.000000, 1000.000000]: NO

Enter a, b, c, d: 13 -17 0 19
Enter u, v: -2 2
[-2.000000, 2.000000]: YES
[-2.000000, 0.000000]: YES
[0.000000, 2.000000]: NO

Enter a, b, c, d: 13 -17 0 19
Enter u, v: -2 0
[-2.000000, 0.000000]: YES
[-2.000000, -1.000000]: YES
[-1.000000, 0.000000]: NO

Enter a, b, c, d: 13 -17 0 19
Enter u, v: -2 -1
[-2.000000, -1.000000]: YES
[-2.000000, -1.500000]: NO
[-1.500000, -1.000000]: YES

Enter a, b, c, d: 13 -17 0 19
Enter u, v: -1.5 -1
[-1.500000, -1.000000]: YES
[-1.500000, -1.250000]: NO
[-1.250000, -1.000000]: YES
```

w03-3 Given a positive integer n (of at most 4 digits), transform it to another number m which is a copy of n, with every prime digit of n (i.e., $2, 3, 5, 7$) made to follow by its next digit (i.e., $2 \mapsto 23, 3 \mapsto 34, 5 \mapsto 56, 7 \mapsto 78$).
Thus, if n has k prime digits, then m will have k digits more than n.
Take n as **`unsigned int`** and all other variables as deemed fit.

Examples

```
Enter n: 2
m = 23.

Enter n: 5
m = 56.

Enter n: 25
m = 2356.
```

```
Enter n: 245
m = 23456.

Enter n: 254
m = 23564.

Enter n: 8520
m = 856230.
```

```
Enter n: 7575
m = 78567856.

Enter n: 7532
m = 78563423.
```

Week 4: 28-April-2021 (Lab Test 1)

w04-1. Declare six integer variables and get their values from the user. Now find the smallest and the largest among these six elements, using as few variables and as few comparisons as possible. You should not use array or any user-defined function or **`math.h`**.
[For no extra variable and fewest comparisons, you get full marks. For every extra variable, you lose 20% marks; and for every extra comparison, you lose 10%.]

Examples

```
Enter six integers: 6 5 1 2 4 3
Min, Max = 1, 6

Enter six integers: 1 2 3 1 4 2
Min, Max = 1, 4

Enter six integers: 13 13 -9 6 -3 5
Min, Max = -9, 13
```

w04-2. Write a program that takes n positive integers as input and finds the sum of their first digits and the sum of their last digits.
You should not use array or any user-defined function or **`math.h`**.

Examples

```
Enter n: 2
Enter the numbers: 2 3
sum = 5, 5.

Enter n: 2
Enter the numbers: 31 2
sum = 5, 3.
```

```
Enter n: 2
Enter the numbers: 2 31
sum = 5, 3.

Enter n: 5
Enter the numbers: 321 756 102 304 506
sum = 19, 19.
```

w04-3. Numbers x_n are defined by the recurrence:

$$x_1 = 2,\ x_n = \frac{1}{2}\left(x_{n-1} + \frac{2}{x_{n-1}}\right) \quad \text{if } n \geq 2.$$

Express the value of each x_n for $n = 2, 3, \ldots, 6$ as a fraction of the form a/b, where a and b are integers and $\mathrm{GCD}(a, b) = 1$. Also print its real value as a floating-point number.
You must use loop but not any array or any user-defined function or **`math.h`**.
Declare n as **`int`** and all other variables as **`unsigned long int`.**
[Can you say what would be the value of $\lim_{n\to\infty} x_n$*? This is not for evaluation.]*

Output:

```
n = 2: 3 / 2 = 1.500000
n = 3: 17 / 12 = 1.416667
n = 4: 577 / 408 = 1.414216
n = 5: 665857 / 470832 = 1.414214
n = 6: 886731088897 / 627013566048 = 1.414214
```

Logic:

Let $x_n = \frac{a}{b}$.

Then $x_{n+1} = \frac{a}{2b} + \frac{b}{a} = \frac{a^2+2b^2}{2ab} \implies a \leftarrow a^2 + 2b^2, b \leftarrow 2ab.$
As $n \longrightarrow \infty,\ x_{n+1} \longrightarrow x_n = k$ (say) $\implies k = \frac{k}{2} + \frac{1}{k} \implies k^2 = 2 \implies k = \sqrt{2}.$

We start with $n = 2 (a = 3, b = 2)$, i.e., $a > 2$, b even, and $\mathrm{GCD}(a, b) = 1$.
We show that $\mathrm{GCD}(a^2 + 2b^2, 2ab) = 1$.
Proof: As b is even, a cannot be even, and so 2 does not divide $a^2 + 2b^2$.
Now, let $p > 2$ be a prime such that p divides either a or b.
If $p|a$, then p cannot divide $a^2 + 2b^2$ because that would imply $p|2b^2$, or $p|b$, which is a contradiction because $\mathrm{GCD}(a, b) = 1$.
If $p|b$, then p cannot divide $a^2 + 2b^2$ because that would imply $p|a^2$, or $p|a$, which is again a contradiction because $\mathrm{GCD}(a, b) = 1$.

Week 5: 19-May-2021 (array, function, recursion)

w05-1. For positive integers n, a function $f(n)$ is defined as

$$f(n) = \begin{cases} 1 & \text{if } n = 1, \\ f(n-1) + n! & \text{otherwise.} \end{cases}$$

Compute the values of $f(1), f(2), \ldots, f(12)$ using fewest multiplications. Store them in an array of size 12. Print the array. Do use loop but not any function.

Output

```
1 3 9 33 153 873 5913 46233 409113 4037913 43954713 522956313
```

w05-2. Write a program to reverse the digits of a number using function. The function should take as input the given number and return to **`main()`** the reversed number.

Examples

```
Enter a positive integer: 10900
Reverse number = 901.

Enter a positive integer: 112453
Reverse number = 354211.
```

w05-3. Write a function **`int nCr(int n, int r)`** that recursively computes the value of $\binom{n}{r}$ using the recurrence

$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$$

and returns the value to **`main()`**. From the **`main()`**, this function has to be called from a loop for $r = 0, 1, 2, \ldots, n$. The value of n is taken as input from **`main()`**.
[This is not an efficient way to compute $\binom{n}{r}$ but it shows how a recursive function works.]

Examples

```
Enter n: 1
Result: 1 1

Enter n: 2
Result: 1 2 1

Enter n: 3
Result: 1 3 3 1

Enter n: 6
Result: 1 6 15 20 15 6 1

Enter n: 10
Result: 1 10 45 120 210 252 210 120 45 10 1
```

Week 6: 02-June-2021 (Lab Test 2)

You can write user-defined functions but cannot use any library function other than **`stdio.h`**.

w06-1. (a) User gives just a positive whole number n as input. Print all the square numbers less than or equal to n. [50 marks]

(b) For each square number s, print the binary string of length s whose first $\sqrt{s}$ bits are 1, next $\sqrt{s}$ bits are 0, next $\sqrt{s}$ bits are 1, and so on. [50 marks]

For the entire code, you cannot use any array or loop but you can use user-defined recursive / non-recursive functions.

Examples

```
Enter a whole number: 20
 1 = 1
 4 = 1100
 9 = 111000111
16 = 1111000011110000

Enter a whole number: 50
 1 = 1
 4 = 1100
 9 = 111000111
16 = 1111000011110000
25 = 1111100000111110000011111
36 = 111111000000111111000000111111000000
49 = 11111110000000111111100000001111111000000001111111
```

w06-2. Input is n distinct digits that should be stored in an array. Output should be all possible numbers made by these digits, and the count of these numbers. The value of n is in the range $[2, 10]$. Every ten numbers should be printed in a new line, excepting possibly the last line. Further, each number needs to be printed right-aligned in its respective column just as a sequence of digits, and hence no extra array can be used.

Examples

```
Enter the number of digits: 2
Enter 2 distinct digits: 5 3
Numbers:
53 35
Count = 2.

Enter the number of digits: 2
Enter 2 distinct digits: 5 0
Numbers:
50  5
Count = 2.

Enter the number of digits: 3
Enter 3 distinct digits: 2 4 6
Numbers:
246 264 426 462 642 624
Count = 6.
```

```
Enter the number of digits: 3
Enter 3 distinct digits: 2 0 6
Numbers:
206 260  26  62 602 620
Count = 6.

Enter the number of digits: 3
Enter 3 distinct digits: 0 2 6
Numbers:
 26  62 206 260 620 602
Count = 6.

Enter the number of digits: 5
Enter 5 distinct digits: 1 3 5 4 2
Numbers:
13542 13524 13452 13425 13245 13254 15342 15324 15432 15423
15243 15234 14532 14523 14352 14325 14235 14253 12543 12534
12453 12435 12345 12354 31542 31524 31452 31425 31245 31254
35142 35124 35412 35421 35241 35214 34512 34521 34152 34125
34215 34251 32541 32514 32451 32415 32145 32154 53142 53124
53412 53421 53241 53214 51342 51324 51432 51423 51243 51234
54132 54123 54312 54321 54231 54213 52143 52134 52413 52431
52341 52314 43512 43521 43152 43125 43215 43251 45312 45321
45132 45123 45213 45231 41532 41523 41352 41325 41235 41253
42513 42531 42153 42135 42315 42351 23541 23514 23451 23415
23145 23154 25341 25314 25431 25413 25143 25134 24531 24513
24351 24315 24135 24153 21543 21534 21453 21435 21345 21354
Count = 120.

Enter the number of digits: 5
Enter 5 distinct digits: 5 4 0 1 2
Numbers:
54012 54021 54102 54120 54210 54201 50412 50421 50142 50124
50214 50241 51042 51024 51402 51420 51240 51204 52014 52041
52104 52140 52410 52401 45012 45021 45102 45120 45210 45201
40512 40521 40152 40125 40215 40251 41052 41025 41502 41520
41250 41205 42015 42051 42105 42150 42510 42501  4512  4521
 4152  4125  4215  4251  5412  5421  5142  5124  5214  5241
 1542  1524  1452  1425  1245  1254  2514  2541  2154  2145
 2415  2451 14052 14025 14502 14520 14250 14205 10452 10425
10542 10524 10254 10245 15042 15024 15402 15420 15240 15204
12054 12045 12504 12540 12450 12405 24015 24051 24105 24150
24510 24501 20415 20451 20145 20154 20514 20541 21045 21054
21405 21450 21540 21504 25014 25041 25104 25140 25410 25401
Count = 120.
```

w06-3. (a) Take as input a natural number n in the range $[1, 100]$, followed by n distinct integers. Store these n numbers in a 1D integer array. Print the n numbers. [30 marks]

(b) Take another integer input m in the range $[1, n]$. Print the smallest m numbers in increasing order. [70 marks]

Examples

```
Enter n: 5
Enter 5 distinct numbers:
6
7
6
Number already exists; please enter a different number.
5
4
4
Number already exists; please enter a different number.
7
Number already exists; please enter a different number.
2
Numbers in the array:
6 7 5 4 2
Enter m: 3
Smallest 3 numbers in order:
2 4 5
```

```
Enter n: 5
Enter 5 distinct numbers:
5
7
6
6
Number already exists; please enter a different number.
5
Number already exists; please enter a different number.
4
8
Numbers in the array:
5 7 6 4 8
Enter m: 5
Smallest 5 numbers in order:
4 5 6 7 8
```

```
Enter n: 10
Enter 10 distinct numbers:
15
11
23
12
62
11
Number already exists; please enter a different number.
```

```
11
Number already exists; please enter a different number.
12
Number already exists; please enter a different number.
11
Number already exists; please enter a different number.
54
45
45
Number already exists; please enter a different number.
11
Number already exists; please enter a different number.
62
Number already exists; please enter a different number.
13
16
14
Numbers in the array:
15 11 23 12 62 54 45 13 16 14
Enter m: 4
Smallest 4 numbers in order:
11 12 13 14
```

Week 7: 9-June-2021 (2D array, structure)

w07-1. Compute the multiplication of two matrices A and B having dimensions (#rows by #columns) m-by-n and n-by-p respectively. The parameters m, n, p (1 to 10) and the elements of A and B are taken as input during run time. Assume that the elements are integers. Compute the product AB, store it in C, and print C on the terminal.

Example 1

```
Enter #rows & #columns of 1st matrix: 1 1
Enter #columns of 2nd matrix: 1
Enter Matrix A:
2
Enter Matrix B:
3
Output matrix:
  6
```

Example 2

```
Enter #rows & #columns of 1st matrix: 1 2
Enter #columns of 2nd matrix: 2
Enter Matrix A:
2 3
Enter Matrix B:
1 0
0 1
Output matrix:
  2   3
```

Example 3

```
Enter #rows & #columns of 1st matrix: 3 2
Enter #columns of 2nd matrix: 2
Enter Matrix A:
1 2
3 4
5 6
Enter Matrix B:
1 0
0 1

Output matrix:
  1   2
  3   4
  5   6
```

Example 4

```
Enter #rows & #columns of 1st matrix: 2 3
Enter #columns of 2nd matrix: 3
Enter Matrix A:
1 2 3
```

```
4 5 6
Enter Matrix B:
1 0 0
0 1 0
0 0 1
Output matrix:
  1   2   3
  4   5   6
```

Example 5

```
Enter #rows & #columns of 1st matrix: 5 3
Enter #columns of 2nd matrix: 2
Enter Matrix A:
2 0 1
0 0 2
1 1 0
0 3 1
1 0 0
Enter Matrix B:
1 2
2 1
1 0
Output matrix:
  3   4
  2   0
  3   3
  7   3
  1   2
```

Example 6

```
Enter #rows & #columns of 1st matrix: 3 5
Enter #columns of 2nd matrix: 2
Enter Matrix A:
-2  0  1 -3  7
 1 -4  5  0 -1
 0  0  2 -2 -3
Enter Matrix B:
 1 -1
-1  0
 2  1
 0 -7
 1  0
Output matrix:
  7  24
 14   4
  1  16
```

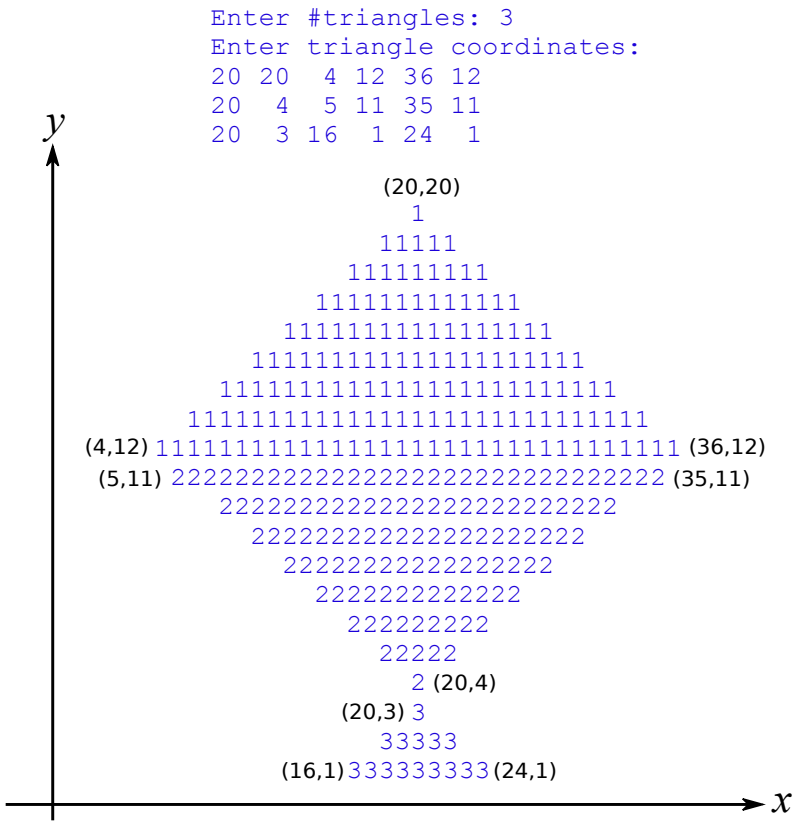
w07-2. Declare the following two structures as global:

i. **`point`** containing two integer coordinates: x, y.

ii. **`triangle`** containing an array of 3 **`point`**s.

Declare in **`main()`** an array of type **`triangle`** that can accommodate 9 triangles. Take the number of triangles n (1 to 9) as input in **`main()`** and populate the array. Assume that the triangles are pairwise disjoint.

Display the n triangles on the terminal as follows. Print each character on the terminal either as the ID (1 to n) of a triangle or as a space, depending on whether that character (suggested to use: $(x, y) \equiv$ (column, row) of the terminal) lies inside some triangle or not. Assume that the vertices of all the triangles have integer coordinates $x \in [0, 40]$, $y \in [0, 20]$.



(x,y) coordinates are shown in black.
Your program has to print only the blue numbers 1, 2, 3.

Note:

i. *Heron's formula:* $\text{area}(\Delta abc) = \sqrt{s(s-a)(s-b)(s-c)}$, where $s = \frac{1}{2}(a+b+c)$, and a, b, c are three sides. We can use Heron's formula to check whether or not a point p (i.e., a terminal character) lies in Δabc, using that p lies in Δabc if and only if

$$\text{area}(\Delta abc) = S, \text{ where } S = \text{area}(\Delta pab) + \text{area}(\Delta pbc) + \text{area}(\Delta pca).$$

However, as a computer has finite precision, this kind of equality check has to be done carefully, using an *error tolerance*, namely ε. So, in your code use that p lies in a triangle Δabc if and only if

$$S - \varepsilon \leq \text{area}(\Delta abc) \leq S + \varepsilon.$$

ii. You can use any other formula to compute triangle area but you must use the above equation to decide whether a point lies inside a triangle.

iii. Use $\varepsilon = 0.001$ for all geometric comparisons.

iv. Use **`double`** precision for real computations whenever needed.

v. Use **`math.h`** for square root function.

vi. Write a user-defined function **`calArea(triangle t)`** to compute the area of a triangle **`t`** as needed in **`main()`**.

Example 1

```
Enter #triangles: 3
Enter triangle coordinates:
20 20  4 12 36 12
20  4  5 11 35 11
20  3 16  1 24  1



Output:

                    1
                  11111
                111111111
              1111111111111
            11111111111111111
          1111111111111111111111
        111111111111111111111111111
      11111111111111111111111111111111
    1111111111111111111111111111111111111
     2222222222222222222222222222222
        22222222222222222222222
          222222222222222222
            2222222222222222
              22222222222222
                222222222
                  22222
                    2
                    3
                  33333
                333333333
```

Example 2

```
Enter #triangles: 7
Enter triangle coordinates:
20 20 16 18 24 18
20 16 16 18 24 18
20 16 13  8 27  8
20 16 10 12  7  7
20 16 30 12 33  7
20  8 15  8 17  0
20  8 25  8 23  0

Output:

                1
              11111
            111111111
              22222
                2
              4 3 5
           444 333 555
         444  33333  555
      44444  3333333  55555
      444   333333333   555
     444   33333333333   555
     4    3333333333333    5
    4    333333333333333    5
   4         6666 7777        5
             6666 7777
             666   777
             666   777
              66   77
              6     7
              6     7
              6     7
```

w07-3. Define a *3D vector* as: **`typedef struct {int x, y, z;} vector;`**
Use it to compute the dot product and cross product of two vectors, using the following user-defined functions, invoking them from **`main()`**, and printing all output from **`main()`**:

i. **`int dot_product(vector a, vector b)`**
ii. **`vector cross_product(vector a, vector b)`**

Note: For two vectors $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$, their respective *dot product* and *cross product* are defined as:
$a \cdot b = a_1b_1 + a_2b_2 + a_3b_3$ and $a \times b = (a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1)$.
The code should be structured as follows.
Note: Any user-defined function should be written before **`main()`** or before a function that invokes it. Henceforth, in all assignments, you should do this; otherwise 20% marks will be deducted.

```
#include<stdio.h>

typedef struct {int x, y, z;} vector;

int dot_product(vector a, vector b){
  // compute the dot product and return the value
}

vector cross_product(vector a, vector b){
  // declare a vector c, compute c as the cross product of a and b
  // return c
}

int main(){
  int x, y, z; // vector components = (x, y, z)
  vector a, b, c;

  // read vectors a and b.
  // call dot_product(a, b) and print its value.
  // call cross_product(a, b) and store it in c.
  // print c.

  return 1;
}
```

Examples

```
Enter vector a: 1 1 1
Enter vector b: 1 2 3
Dot product = 6.
Cross product = (1, -2, 1).

Enter vector a: -2 3 -4
Enter vector b: 4 -5 6
Dot product = -47.
Cross product = (-2, -4, -2).
```