name of file : filename.hs

.hs files are compiled and run

---

helloworldpgm: with newline in the end

```
main = putStrLn "Hello, World!"
```

output: Hello, World!

without newline in the end

```
main = putStr "Hello, World!"
```

output: Hello, World!

---

Area of square:

```
main = do
    let x = 10
    putStr”The area in sq mts is: “
    print(x*x)
```

Area of triangle, perimeter of rectangle, volume of sphere

---

get is used for inputing strings for eg

```
x <- getLine
l <- getLine
b <- getLine
```

Theses strings have to be read as int / float etc for calculations

---

Lists :

```
let list1 = [1,3,5,7,9]
print list1
```

to print summation of elements of the list:

```
print( sum (list1))
print( sum (filter odd (list1) )) {- Summation of odd Nos-}
print( sum (filter even (list1) )) {- Summation of even Nos-}
```

Recursion in Haskell: To compute sum of 0 to n

```
func1 n | n==0=0
func1 n | n/=0=n+func1(n-1) {- /= is not equal to -}
main = do print(func1 6)
```

---

product of elements of lists:

Cartesian Product:

```
[x*y | x <- [1,2,3], y <- [9,8,7]]
[9,8,7,18,16,14,27,24,21]
```

---

If desired answer is : [1*9,2*8,3*7]

```
prslst [] [] z = reverse z
prlst (x:xs) (y:ys) z = prslst xs ys ((x*y):z)
main = print(prslst [1,2,3] [9,8,7] [])
```

---

Representation of multidimensional matrices as multidimensional arrays.

A = 1,2,3 B = -2,1,3 C = A X B = 1*-2 + 2*1 + 3*3 = 9, 1*1 + 2*4 + 3*2 = 15

4,5,6 1,4,2

1,2,0 3,2,1

In the machine, the address of 4 is 1 unit + address of 3. I.e. address space is linearised.

***Try a c program and display address in %p format of all elements of a 2D or 3D array.***

in Prolog and Haskell data structure to be used is lists.

given A = [1,2,3,4,5,6,1,2,0] and B = [-2,1,3,1,4,2,3,2,1]

A is represented as [1,2,3,4,5,6]

multnum should calculate partial sums of product of x with all elements of y a

*C = A + B for matrix do pairwise addition in way we did pairwise multiplication*

```
mulrow (x:xs) y z = (multnum x y  : z )
```