1. C

1.1. testC++.

```cpp
#include <stdio.h>
#include <vector>
#include <cstring>
#include <algorithm>

using namespace std;

int main()
{
    int x[2];
    int T;
    int len[3][2];

    scanf(" %d",&T);

    while(T--) {
        for (int i = 0; i < 3; i++ ) {
            scanf(" %d %d", &x[0], &x[1]);

            if (x[0] >= 0 && x[1] >= 0) {
                len[i][0] = x[1]-x[0];
                len[i][1] = 0;
            } else if (x[0] < 0 && x[1] >= 0) {
                len[i][0] = x[1];
                len[i][1] = -x[0];
            } else {
                len[i][0] = 0;
                len[i][1] = x[1]-x[0];
            }
        }

        printf("%d %d %d %d %d %d %d %d\n"
            , len[0][0]*len[1][0]*len[2][0]
            , len[0][1]*len[1][0]*len[2][0]
            , len[0][1]*len[1][1]*len[2][0]
            , len[0][0]*len[1][1]*len[2][0]
            , len[0][0]*len[1][0]*len[2][1]
            , len[0][1]*len[1][0]*len[2][1]
            , len[0][1]*len[1][1]*len[2][1]
            , len[0][0]*len[1][1]*len[2][1]);
    }

    return 0;
}
```

2. Java

2.1. testJAVA.

```java
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

3. Python

3.1. testPython.

```python
from math import cos, pi, ceil, sin, sqrt

def compute(P, idx):
  F = complex(0, 0)
  for i, p in enumerate(P):
    if i == idx: continue
    f = 1 / pow(abs(p-P[idx]), 3);
    F += f * (p-P[idx]);
  return F


R = [i*0.01 for i in range(1,10)]
dL = 0.003
dt = 1
G = 6.67e-11
m = 1
pos = []
vel = []

for r in R:
  Len = 2*r*pi
  N = ceil(Len/dL)
  pp = [r*complex(cos(i/N*2*pi), sin(i/N*2*pi)) for i in
      range(N)]
  L = len(pos)
  pos.extend(pp)
  for i in range(L, len(pos)):
    F = G*m*m*compute(pos, i)
    A = dt*dt*((pos[i].real*pos[i].real)+(pos[i].imag*pos[i
        ].imag))
    B = 2*dt*(pos[i].real*(dt*dt*F.imag/m+pos[i].imag)-pos[i
        ].imag*(dt*dt*F.real/m+pos[i].real))
    C = (dt*dt*F.imag/m+pos[i].imag)*(dt*dt*F.imag/m+pos[i].
        imag) + (dt*dt*F.real/m+pos[i].real)*(dt*dt*F.real/m
        +pos[i].real) - r*r
    k = (-B + sqrt(B*B - 4*A*C))/(2*A)

```

```
33      vel.append(complex(pos[i].imag*(-k), pos[i].real*k)-2*
            complex(0,5172*sqrt(G/0.65/(945+5172))))
34
35 print(len(pos))
36 for i in range(len(pos)):
37   print(pos[i].real+0.65,pos[i].imag,vel[i].real,vel[i].imag
         ,sep=" ")
```