

# Neps Factory



Na fábrica **Neps Factory** existem várias caixas empilhadas em 3 pilhas diferentes (conforme a figura abaixo). Normalmente os empregados tem que manualmente remover as caixas quando precisam ser transportadas para a linha de produção. Porém, buscando ser uma fábrica mais moderna, a **Neps Factory** resolveu experimentar o uso de robôs.

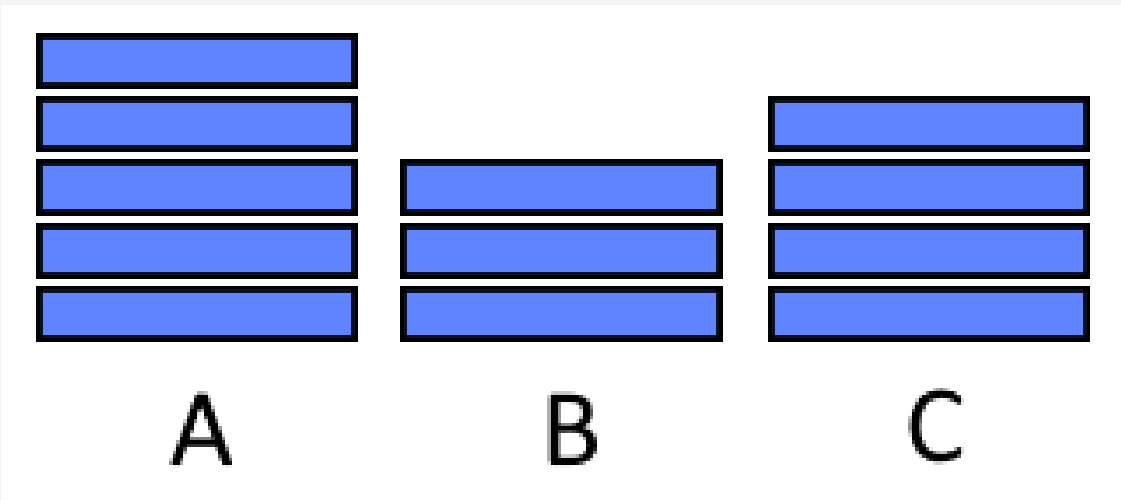


Figura 1

Os robôs vem em dois modelos diferentes, cada modelo tem uma estratégia diferente para transportar as caixas. A cada tempo  $t$  os robôs fazem as seguintes operações:

- Modelo A: Transporta 1 caixa de cada pilha.
- Modelo B: Transporta 3 caixas da pilha A, caso não tenha mais caixas na pilha A, transporta 3 caixas da pilha B, caso não haja caixas em B, transporta 3 caixas da pilha C.

A fábrica resolveu criar um programa que simula o comportamento dos dois roboreos, assim ficará mais fácil decidir entre qual modelo comprar.

O código abaixo simula dois roboreos, um de cada modelo competindo para ver quem consegue completar a tarefa primeiro. Porém o código da classe ModeloA e ModeloB estão faltando, sua tarefa é implementar ambas as classes.

```
1  #include <stdio.h>
2
3  class Pilhas{
4      int a, b, c;
5  public:
6
7      Pilhas(int a, int b, int c){
8          this->a = a;
9          this->b = b;
10         this->c = c;
11     }
12
13     int get_a(){ return a; }
14
15     int get_b(){ return b; }
16
17     int get_c(){ return c; }
18
19     void remover_caixas(int a, int b, int c){
20         this->a = this->a - a > 0 ? this->a - a : 0;
21         this->b = this->b - b > 0 ? this->b - b : 0;
22         this->c = this->c - c > 0 ? this->c - c : 0;
23     }
24
25     bool todas_vazias(){
26         if (this->a == 0 and this->b == 0 and this->c == 0){
27             return true;
28         }
29
30         return false;
31     }
32 };
33
34 class Robo {
35 protected:
36     bool completou;
37 public:
38     bool completou_tarefa(){ return this->completou; }
39     virtual void operar(Pilhas &P)=0;
40 };
41
42 //TODO: Implementar classe ModeloA que herda da classe Robo.
43 //TODO: Implementar classe ModeloB que herda da classe Robo.
44
45 int main(){
46
47     Robo *modeloA;
48     Robo *modeloB;
49     modeloA = new ModeloA();
50     modeloB = new ModeloB();
51
52
53     int a, b, c;
54     scanf("%d %d %d", &a, &b, &c);
55
56     Pilhas PA = Pilhas(a, b, c);
57     Pilhas PB = Pilhas(a, b, c);
58
59     int i = 0;
60     while( !modeloA->completou_tarefa() and !modeloB->completou_tarefa() ){
61         modeloA->operar(PA);
62         modeloB->operar(PB);
63     }
64
65     if(modeloA->completou_tarefa() and modeloB->completou_tarefa()){
66         printf("EMPATE");
67     }else if (modeloA->completou_tarefa()){
68         printf("MODELO A");
69     }else {
70         printf("MODELO B");
71     }
72
73 }
```

OBS: A linha 20 do código acima é equivalente ao código abaixo:

```
1  if (this->a - a > 0){
2      this->a -= a;
3  }else{
4      this->a = 0;
5  }
```

## Entrada

A entrada do seu programa será uma linha contendo 3 inteiros. A quantidade de caixas na pilha A, B e C, respectivamente.

## Saída

A saída do seu programa deve imprimir qual modelo de robô completou a tarefa primeiro ou "EMPATE" caso os dois completem ao mesmo tempo.

## Restrições


- A quantidade de caixas em uma pilha pode variar entre 1 e 100.


Exemplos de Entrada	Exemplos de Saída
3 0 0	MODELO B
3 3 3	EMPATE
5 5 5	MODELO A


Traduzido por **Luis Paulo**

## Detalhes

 **ESCREVER SOLUÇÃO**

 SUBMISSÕES

 COMUNIDADE

 ANOTAÇÕES

 NÃO HÁ UM TUTORIAL

**Tempo Limite:** 1 second(s)

**Limite de Memória:** 256 mb

**XP:** 40

**Cronômetro:** 

**Adicionado por :** **Thiago Nepomuceno**

**Resolvido por:** 38 usuários

**No Neps desde:** 09/04/2020

