Neste desafio a maior dificuldade foi fazer com que todos os métodos fossem executados em O (1) . A variavel **minEle** variável armazena o elemento mínimo atual na pilha. Agora, a questão que me fez quebrar a cabeça foi a seguinte, mas e se o elemento mínimo atual for removido?

Para lidar com isso, usaremos "2*minEle - t" na pilha, para que o elemento mínimo anterior possa ser recuperado usando o minEle atual e seu valor armazenado na pilha.

```
#include <bits/stdc++.h>
using namespace std;

// Uma pilha definida pelo usuário que suporta getMin()
// adição para push() e pop ()
struct Pilhas
{
 stack<int> voxus;
        int minEle;

// Imprime elemento mínimo da Pilha
        void getMin()
        {
                if (voxus.empty())
                        cout << "Pila atual vazia\n";

        // minEle variável armazena o elemento mínimo
// na pilha.
                else
                        cout <<"Elemento mínimo na pilha é: "
                                << minEle << "\n";
        }

// Imprime o elemento "top" na Pilha
```

```
void peek()
	{
		if (voxus.empty())
		{
			cout << "Pilha Vazia ";
			return;
		}

		int t = voxus.top(); // Elemento "TOP"

		cout << "Elemento do Top é: ";

	// Se t <minEle significa que minEle armazena
   // valor de t.
		(t < minEle)? cout << minEle: cout << t;
	}

// Remova o elemento superior (top) da pilha
void pop()
	{
		if (voxus.empty())
		{
			cout << "Pilha vazia\n";
			return;
		}

		cout << "Top elemento removido: ";
		int t = voxus.top();
		voxus.pop();
```

```
        if (t < minEle)
        {
                cout << minEle << "\n";
                minEle = 2*minEle - t;
        }

        else
                cout << t << "\n";
}

// Remover elemento "top" da pilha
void push(int x)
{
        // inserindo um novo numero na pilha
        if (voxus.empty())
        {
                minEle = x;
                voxus.push(x);
                cout << "Numero inserido: " << x << "\n";
                return;
        }

        // Se o novo número for menor que minEle
        if (x < minEle)
        {
                voxus.push(2*x - minEle);
                minEle = x;
        }
```

```
            else
            voxus.push(x);

            cout << "Numero inserido: " << x << "\n";
        }
};

int main()
{
        Pilhas voxus;
        voxus.push(3);
        voxus.push(5);
        voxus.getMin();
        voxus.push(2);
        voxus.push(1);
        voxus.getMin();
        voxus.pop();
        voxus.getMin();
        voxus.pop();
        voxus.peek();

        return 0;
}
```

Abaixo estão detalhadas etapas e explicações sobre o trabalho.

Push (x) : Insere x na parte superior da pilha.

- Se a pilha estiver vazia, insira x na pilha e faça minEle igual a x.
- Se a pilha não estiver vazia, compare x com minEle. Surgem dois casos:
    - Se x for maior ou igual a minEle, basta inserir x.

- Se x for inferior a minEle, insira (2 * x - minEle) na pilha e faça minEle igual a x. Por exemplo, deixe o minEle anterior ser 3. Agora queremos inserir 2. Atualizamos o minEle como 2 e inserimos 2 * 2 - 3 = 1 na pilha.

Pop (): Remove um elemento do topo da pilha.

- Remova o elemento do topo. Deixe o elemento removido ser y. Surgem dois casos:
  - Se y for maior ou igual a minEle, o elemento mínimo na pilha ainda é minEle.
  - Se y for inferior a minEle, o elemento mínimo agora se torna (2 * minEle - y), portanto, atualize (minEle = 2 * minEle - y). É aqui que recuperamos o mínimo anterior do mínimo atual e seu valor na pilha. Por exemplo, deixe o elemento ser removido seja 1 e minEle seja 2. Removemos 1 e atualizamos o minEle como 2 * 2 - 1 = 3.

Observações:

- A pilha não possui o valor real de um elemento se for mínimo até esse momento.
- O elemento mínimo real é sempre armazenado em minEle

Push(X)

Numero inicial	PILHA ATUAL	minELE
3	3	3
5	5 3	3
2	1 5 3	2
1	0 1 5 3	1
1	1 0 1 5 3	1
-1	-3 1 0 1 5 3	-1

- Número a ser inserido: 3, a pilha está vazia, então insira 3 na pilha e minEle = 3.
- Número a ser inserido: 5, a pilha não está vazia, 5> minEle, insira 5 na pilha e minEle = 3.
- Número a ser inserido: 2, a pilha não está vazia, 2 <minEle, insira (2 * 2-3 = 1) na pilha e minEle = 2.
- Número a ser inserido: 1, a pilha não está vazia, 1 <minEle, insira (2 * 1-2 = 0) na pilha e minEle = 1.
- Número a ser inserido: 1, a pilha não está vazia, 1 = minEle, insira 1 na pilha e minEle = 1.
- Número a ser Inserido: -1, a pilha não está vazia, -1 <minEle, insira (2 * -1 - 1 = -3) na pilha e minEle = -1.

Pop(Y)

Numero removido	Numero Original	Pilha atual	minELE
-1		-3 1 0 1 5 3	-1
-3	-1	1 0 1 5 3	1
1	1	0 1 5 3	1
0	1	1 5 3	2
1	2	5 3	3
5	5	3	3

- Inicialmente, o elemento minEle mínimo na pilha é -1.
- Número removido: -3, uma vez que -3 é inferior ao elemento mínimo, o número original que está sendo removido é minEle que é -1 e o novo minEle = 2 * -1 - (-3) = 1
- Número removido: 1, 1 == minEle, então o número removido é 1 e minEle ainda é igual a 1.
- Número removido: 0, 0 <minEle, o número original é minEle, que é 1 e novo minEle = 2 * 1 - 0 = 2.
- Número removido: 1, 1 <minEle, o número original é minEle que é 2 e novo minEle = 2 * 2 - 1 = 3.
- Número removido: 5, 5> minEle, o número original é 5 e minEle ainda é 3