CS1103

Programación Orientado a Objetos II

Práctica Calificada #1 (PC1)

Sección 101

2020 - 2

Profesor: Rubén Rivas

Nombre: _____

5 puntos

1. Desarrollar la función **roll** que permita rotar hacia la derecha los elementos de una lista enlazada. El primer parámetro de la función es un **utec::linked_list** de tipo genérico y el segundo parámetro un valor entero **n** que representa el desplazamiento de sus elementos hacia la derecha, la función **roll** deberá de retornar una lista enlazada donde los elementos de la primera lista se encuentren rotados las posiciones definidas por el segundo parametro,.

   Declaración de la función:

```
using namespace utec;

    template<typename T>
    linked_list<T> roll(linked_list<T>& lista, int desplazar);
```

   Ejemplo #1:

```
    utec::linked_list<int> lista = {1, 2, 3, 4, 5, 6};
    auto resultado = roll(lista, 2);
    // resultado contendrá {5, 6, 1, 2, 3, 4}.
```

   Ejemplo #2:

```
    utec::linked_list<double> lista = {10.1, 20, 40.0, 50.3};
    auto resultado = roll(lista, 7);
    // resultado contendrá {40.0, 50.3, 10.1, 20, 40}.
```

5 puntos

2. Desarrollar la función **filter** que tenga como primer parámetro un contenedor que podrías ser de cualquier tipo secuencial (std::vector, std::array, std::list, std::forward_list o std::deque) y que tenga un segundo parámetro del tipo genérico (variadic) y que permita definir las posiciones que deberán de copiarse hacia el segundo container:

   Ejemplo #1:

```
std::vector vec = {1, 3, 4, 5, 6, 7};
auto r1 = filter(vec, 5, 2, 4);
// r1 contendrá {7, 4, 6}.
```

   Ejemplo #2:

```
std::list lst = {1, 3, 4, 5};
auto r2 = filter(lst, 2, 0);
// r2 contendrá {4, 1}.
```

   Ejemplo #1:

```
std::forward_list fwd_lst = {1, 3, 4, 5, 6, 7};
auto r3 = filter(fwd_lst, 3, 2, 1);
// r3 contendrá {5, 4, 3}.
```

5 puntos

3. Escribir una función **calculate** que reciba 3 parámetros el primer parámetro debe ser del tipo std::array, std::vector, std::list o std::deque y el segundo parámetro un std::map cuyo tipo de key podría ser de cualquier tipo y cuyo tipo de valor debe coincidir con el tipo base del primer parámetro o viceversa el primer parámetro del tipo std::map y el segundo de tipo std::array, std::vector, std::list o std::deque.
   Adicionalmente debe incluir un 3er parámetro que debe ser un parámetro que soporte una lambda o callable con 2 parámetros que coincidan con el tipo base del primer parámetro o del segundo según el caso y que permita realizar una operación, devolviendo un nuevo contenedor del tamaño de parámetro de mayor tamaño y del tipo vector por defecto, que contenga los valores resultado de la operación definida en el callable.

   Ejemplo:

```
std::vector<int> v1 = {1, 2, 3, 10, 20, 15};
std::map<char, int> m1 = {{'A',1}, {'B',1}, {'C',1}, {'D',1}, {'E',1}};

auto resultado = calculate (v1, m1, [] (int a, int b) {return a + b;});
// resultado contendrá {2, 3, 4, 11, 21, 15}
```

5 puntos

4. Determinar y justificar el Big O de la función f1:

```
template <typename T>
auto random(T first, T last) {
    std::random_device device;
    std::default_random_engine engine(device);
     if constexpr (std::is_integral<T>::value) {
         std::uniform_int_distribution<T> distribution(first, last);
         return distribution(engine);
     }
     else {
         std::uniform_real_distribution<T> distribution(first, last);
         return distribution(engine);
     }
}

int f1(int x) {
    if (x <= 0) return 0;
    else {
        auto i = random(0, x - 1);
        return f1(i) + f1(x - 1 - i);
    }
}
```

Barranco, 14 de octubre del 2020