CS1103

Programación Orientado a Objetos II

Práctica Calificada #2 (PC2)

Sección 201

2021 - 1

Profesor: Rubén Rivas:

6.5 puntos: Heap

1. Escribir la función **join_sort** que procese un número variado de contenedores secuenciales ordenados y que retorne de forma eficiente un contenedor con todos los valores ordenados.

```
// contenedores
vector <int> v1 = {2, 3, 4};
vector <int> v2 = {3, 4, 6, 11};
vector <int> v3 = {6, 7, 9, 13};
vector <int> v4 = {5, 8, 10};
vector <int> v5 = {1, 3, 5};
vector <vector <int>> vj = {v1, v2, v3, v4, v5};
// sorted number
auto res = join_sort (vj);
// muestra el resultado
copy(begin(res), end(res), istream_iterator<int>(cout, " "));
// 1 2 3 3 3 4 4 5 5 6 6 7 8 9 10 11 13
```

```
// contenedores
vector <int> v1 = {3, 4};
vector <int> v2 = {3, 4, 6, 11};
vector <int> v3 = {2, 7, 9, 13};
vector <int> v4 = {4, 8, 10};
vector <int> v5 = {5};
// buscar rango
auto res = join_sort ({v1, v2, v3, v4, v5});
// muestra el resultado
copy(begin(res), end(res), istream_iterator<int>(cout, " "));
// 2 2 2 4 4 4 5 6 7 8 9 10 11 12
```

6.5 puntos: hash

Escribir la función **focus_values**, que permita a partir de un contenedor generar un nuevo contenedor que contenga los valores originales de modo que sean reordenados colocando todos los valores iguales a partir de la posición de su primera ocurrencia.

Ejemplo #1:

```
vector<int> vec1 = {11, 12, 3, 4, 15, 8, 11, 4, 3, 3 12, 8, 9};
auto res = focus_values (vec1);
copy(begin(res), end(res), istream_iterator<int>(cout, " "));
cout << endl;
// 11 11 12 12 3 3 4 4 15 8 8 9
```

Ejemplo #2:

```
vector<int> vec1 = {9, 10, 2, 5, 8, 4, 3, 1, 1, 10, 5, 7, 4};
auto res = focus_values (vec1);
copy(begin(res), end(res), istream_iterator<int>(cout, " "));
cout << endl;
// 9 10 10 2 5 5 8 4 4 3 1 1 7
```

Ejemplo #3:

```
deque<double> deq1 = {5.5, 1, 2, 13, 4, 17, 8, 29, 40.5, 17, 13};
auto res = focus_values (deq1);
copy(begin(res), end(res), istream_iterator<int>(cout, " "));
cout << endl;
// 5.5 1 2 13 13 4 17 17 8 29 40.5
```

5 puntos: Binary Tree

Escribir la función **calculate_distance** que permite calcular la distancia entre 2 nodos.

Ejemplo #1:

```
binary_tree<int> bt1(10);
// Izquierda
auto left_branch = bt1.add_left(bt1.get_root(), 8);
bt1.add_left(left_branch, 3);
auto first = bt1.add_right(left_branch, 4);
// Derecha
auto right_branch = bt1.add_right(bt1.get_root(), 7);
bt1.add_left(right_branch, 8);
auto second = bt1.add_right(right_branch, 4);
// Verificar si es heap
cout << calculate_distance (first, second) << endl;  // 4
```

Ejemplo #2:

```
binary_tree<int> bt2(20);
// Izquierda
auto left = bt2.add_left(bt2.get_root(), 12);
bt2.add_left(left, 5);
bt2.add_right(left, 8);
// Derecha
auto right = bt2.add_right(bt1.get_root(), 7);
bt2.add_left(right, 4);
// Verificar si es heap
cout << calculate_distance (left, right) << endl; // 2
```

Barranco, 13 de junio del 2021