1. Determine the output:

Mtrue  
MM§Mfalse  
MM§Mfalse  
MM§Mtrue  
MM§M123  
MM§M4

2. Create a method that takes a HashMap<Integer, Integer> and returns the sum of the keys of the HashMap.

public <K, V extends Number> double sumKeys(Map<K, V> map) {

double sum = 0.0d;

for (Number key : map.key())

sumkey += key.doubleValue();

return sumkey;

}

3. Create a method that takes a HashMap<Integer, Integer> and returns the sum of all keys and values of the HashMap.

For example, if the input is [1=9, 3=6, 4=9, 6=8, 7=6] then the method should return 59.

Public sumKeyValue(Map<K, V> map)

{

public <K, V extends Number> double sumKeys(Map<K, V> map)

{

double sum = 0.0d;

for (Number key : map.key())

sumkey += key.doubleValue();

return sumkey;

}

public <K, V extends Number> double sumValues(Map<K, V> map)

{

double sum = 0.0d;

for (Number value : map.values())

sumvalue += value.doubleValue();

return sumvalue;

}

return sumkey+sumvalue;

}

4. Create a method that takes a HashMap<String, Integer> and removes all mappings with value less than or equal to 0.

For example, if the input is ["abc"=5, "qwe"=0, "asd"=7, "zxc"=9, "iop"=-2], then the resulting HashMap should be

["abc"=5, "asd"=7, "zxc"=9].

unordered\_map<string, int> remove(unordered\_map<string, int> umap)

{

  for (auto x : umap)

{

if(x.second<1)

{

umap.erase(x.first);

}

}

for (auto x : umap)

{

cout << x.first << " " << x.second << endl;

}

return umap;

}