```cpp
/* Miguel Antão Pereira Amaral 78865 */
/* Aula Quarta feira 15h 30 */
#include <iostream>
#include <map>
#include <utility>
#include <forward_list>
#include <list>

class Cell {
private:
  std::pair <int,int> _coords;
public:
  Cell(int x, int y){
    _coords = std::make_pair (x,y);
  }

  int getCoordinateX(){
    return _coords.first;
  }
  int getCoordinateY(){
    return _coords.second;
  }

  std::pair <int,int> getCoords(){
    return _coords;
  }

  virtual int getValue() =0;
  friend bool operator<( Cell& l,  Cell& r) {
       return l.getValue() < r.getValue();
   }
  friend bool operator>( Cell& l,  Cell& r) {
       return r < l;
   }
  virtual void print(std::ostream& where)  =0 ;
  friend std::ostream &operator<<(std::ostream &os, Cell   &cell) {
    cell.print(os);
    return os;
  }
};

class IntCell : public Cell {
private:
  int _element=0;
public:
  IntCell(int x, int y, int value): Cell(x,y){
    _element = value;
  }
  void setValue(int newValue){
    _element = newValue;
  }
  int getValue(){
    return _element;
  }
  int getElement(){
    return _element;
  }
  void print(std::ostream& os) {
    os << _element;
  }
};

class StringCell : public Cell {
private:
  std::string _element="";
public:
  StringCell(int x, int y, std::string string): Cell(x,y){
    _element = string;
  }
  int getValue(){
    return 0;
  }
  std::string getElement(){
    return _element;
  }
```

```
 76   void print(std::ostream& os) {
 77     os << _element;
 78   }
 79 };
 80
 81 class RefCell : public Cell {
 82 private:
 83   std::shared_ptr<Cell> _element;
 84 public:
 85   RefCell(int my_x, int my_y, std::shared_ptr<Cell> cell): Cell(my_x,my_y){
 86     _element = cell;
 87   }
 88   int getValue(){
 89     return _element->getValue();
 90   }
 91   void print(std::ostream& os) {
 92     os << "REF TO --> [x: " << getCoordinateX() << " | y: " ;
 93     os << getCoordinateY() << "]: ";
 94     os << *_element;
 95   }
 96 };
 97
 98 class FormulaCell : public Cell {
 99 private:
100   std::list< std::shared_ptr<Cell> > _mylist;
101 public:
102   FormulaCell(int my_x, int my_y, std::list< std::shared_ptr<Cell> > lista)\
103                                                                 : Cell(my_x,my_y){
104     _mylist = lista;
105   }
106   int getValue(){
107     int soma =0;
108     for (std::shared_ptr<Cell> x: _mylist){
109       soma += x->getValue();
110     }
111     return soma;
112   }
113   void print(std::ostream& os) {
114     os << "Formula ( ";
115     int i=0;
116     for (std::shared_ptr<Cell> x: _mylist){
117       os << "arg nr:" << i << " " << *x << " | ";
118       i++;
119     }
120     os << ") ";
121   }
122 };
123
124 class CalcSheet {
125 private:
126   std::map<std::pair<int ,int>, std::shared_ptr<Cell> > _myMap;
127 public:
128   std::shared_ptr<Cell> getCell(std::pair <int,int> pair){
129     std::map<std::pair <int,int>, std::shared_ptr<Cell> >::iterator it;
130     it = _myMap.find(pair);
131     if (it == _myMap.end())
132       std::cout << "\e[31m[ERROR] \e[0mCell not found" << std::endl;
133     return it->second;
134   }
135
136   void addCell(std::shared_ptr<Cell> cell){
137     std::pair<int,int> coords = cell->getCoords();
138     std::pair<std::map<std::pair <int,int>, std::shared_ptr<Cell> >   \
139                                                         ::iterator,bool> it;
140     std::pair<std::pair <int,int>, std::shared_ptr<Cell> > toInsert = \
141                                                 std::make_pair (coords,cell);
142
143     it = _myMap.insert (toInsert);
144     if (it.second==false) {
145       std::cout << "\e[31m[ERROR] \e[0mAs coordenas [x=" << coords.first
146       << " | y=" << coords.second <<"] ja estao em uso" << std::endl;
147     }
148   }
149
150   friend std::ostream &operator<<(std::ostream &os, CalcSheet &folha) {
```

```
151     os << "\e[33m[INFO]\e[0m printing CalcSheet" << std::endl;
152     for (auto& elem : folha._myMap){
153       std::shared_ptr<Cell> c1 = elem.second;
154 
155       os << "[x: " << c1->getCoordinateX() << " ";
156       os << "| y: " << c1->getCoordinateY() << "]: ";
157       os <<*c1 << std::endl;
158     }
159     os << "\e[33m[INFO]\e[0m --end" << std::endl;
160     return os;
161   }
162 };
163 
164 int main(){
165     std::list< std::shared_ptr<Cell> > lista;
166     std::list< std::shared_ptr<Cell> > lista2;
167     std::shared_ptr<CalcSheet> folha = std::make_shared<CalcSheet>();
168     std::shared_ptr<IntCell> c1 = std::make_shared<IntCell>(0,0,5);
169     std::shared_ptr<StringCell> c2 = std::make_shared<StringCell>(0,1,"string");
170     std::shared_ptr<RefCell> c3 = std::make_shared<RefCell>(0,2,c1);
171     std::shared_ptr<RefCell> c4 = std::make_shared<RefCell>(0,3,c2);
172     std::shared_ptr<IntCell> c5 = std::make_shared<IntCell>(0,4,5);
173     std::shared_ptr<StringCell> c6 = std::make_shared<StringCell>(0,5,"Tareco");
174 
175     lista.push_front(c3);
176     lista.push_front(c4);
177     lista.push_front(c5);
178     std::shared_ptr<FormulaCell> c7 = \
179                 std::make_shared<FormulaCell>(2,2,lista);
180     lista2.push_front(c1);
181     lista2.push_front(c7);
182     lista2.push_front(c2);
183     lista2.push_front(c3);
184     std::shared_ptr<FormulaCell> c8 = \
185                 std::make_shared<FormulaCell>(1,1,lista2);
186 
187     std::cout << "c7 value: " << c7->getValue() << std::endl;
188     std::cout << "c8 value: " << c8->getValue() << std::endl;
189     std::cout << "c1: " << *c1 << std::endl;
190     std::cout << "c2: " << *c2 << std::endl;
191     std::cout << "c3: " << *c3 << std::endl;
192     std::cout << "c4: " << *c4 << std::endl;
193 
194     c1->setValue(11);
195     folha->addCell(c1);
196     folha->addCell(c2);
197     folha->addCell(c1);
198     folha->addCell(c3);
199     folha->addCell(c4);
200     folha->addCell(c5);
201     folha->addCell(c6);
202     folha->addCell(c7);
203     folha->addCell(c8);
204     std::cout << *folha;
205     c1->setValue(22);
206     std::cout << *folha;
207 
208     bool comparacao = *c1 < *c2;
209     std::cout << "c1: " << *c1 << " < c2: " << *c2
210       << " ? answer: " << comparacao << std::endl;
211 
212     comparacao = *c2 < *c1;
213     std::cout << "c2: " << *c2 << " < c1: " << *c1
214       << " ? answer: " << comparacao << std::endl;
215 
216     comparacao = *c5 < *c1;
217     std::cout << "c5: " << *c5 << " < c1: " << *c1
218       << " ? answer: " << comparacao << std::endl;
219 }
220 
```