90
(девяносто)
(Корныхин Е.В.)

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. М.В.ЛОМОНОСОВА * ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ *
ПРИЕМНАЯ КОМИССИЯ

РОБОФЕСТ

Олимпиада школьников

«РОБОФЕСТ-2016»

Заключительный этап по информатике

Фамилия Имя Отчество участника Никифоров Никита Игоревич

Класс 10 Школа 2086

Дата 16.04.16 Подпись участника

Шифр robo16_031.

--- Task A ---

--- Solution 48-000000-34206-robo16_031-robo16_031-20160416153047.cpp ---

```
#include <iostream>
#include <vector>
#include <utility>
#include <set>

int h, w;
std::vector<char> field;


int num(int x, int y){
  return x + y*w;
}

char get(int x, int y){
  if(x < 0 or x >= w)return 'b';
  if(y < 0 or y >= h)return 'b';

  return field[num(x, y)];
}

std::vector<char> gen_algorm(int x, int y, std::set<std::pair<int, int>> was){
  was.insert({x, y});
  std::vector<char> some;
  bool first = true;
  if(get(x, y + 1) == '.' and !was.count({x, y + 1})){
    std::vector<char> some_now=gen_algorm(x, y + 1, was);
    if(some_now.size() != 0){
      some_now.push_back('U');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x, y - 1) == '.' and !was.count({x, y - 1})){
    std::vector<char> some_now=gen_algorm(x, y - 1, was);
    if(some_now.size() != 0){
      some_now.push_back('D');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x + 1, y) == '.' and !was.count({x + 1, y})){
    std::vector<char> some_now=gen_algorm(x + 1, y, was);
    if(some_now.size() != 0){
      some_now.push_back('R');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x - 1, y) == '.' and !was.count({x - 1, y})){
    std::vector<char> some_now=gen_algorm(x - 1, y, was);
    if(some_now.size() != 0){
      some_now.push_back('L');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(some.size() != 0)return some;
  if(get(x, y + 1) == 'E'){
    some.push_back('U');
    return some;
  }
  if(get(x, y - 1) == 'E'){
    some.push_back('D');
    return some;
  }
  if(get(x + 1, y) == 'E'){
    some.push_back('R');
    return some;
  }
  if(get(x - 1, y) == 'E'){
    some.push_back('L');
    return some;
  }
  return some;
}

std::pair<int, int> find_robot(){
  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      //std::cout << x << " "<< y << " " << field[num(x, y)] << std::endl;
      if(field[num(x, y)] == '@')return {x, y};
    }
  }
  return {-1, -1};
}


int main(){

  std::cin >> w >> h;

  std::vector<char> fd;

  for(int i =0;i < w*h; i++){
    char now;
    std::cin >> now;
    fd.push_back(now);
  }
  for(int y = h - 1; y >= 0; y--){
    for(int x = 0; x < w; x++){
      field.push_back(fd[num(x, y)]);
    }
  }


  std::pair<int, int> now_map = find_robot();
  //std::cout << now_map.first << " " << now_map.second << std::endl;
  std::set<std::pair<int, int>> was;
  auto commands = gen_algorm(now_map.first, now_map.second, was);

  for(int i = commands.size() - 1; i >= 0; i--){
    std::cout << commands[i];
  }
  std::cout << std::endl;
}
```

--- Solution 48-000001-34206-robo16_031-robo16_031-20160416153232.cpp ---

```
#include <iostream>
#include <vector>
#include <utility>
```

```
#include <set>

int h, w;
std::vector<char> field;


int num(int x, int y){
  return x + y*w;
}

char get(int x, int y){
  if(x < 0 or x >= w)return 'b';
  if(y < 0 or y >= h)return 'b';

  return field[num(x, y)];
}

std::vector<char> gen_algorm(int x, int y, std::set<std::pair<int, int>> was){
  was.insert({x, y});
  std::vector<char> some;
  bool first = true;
  if(get(x, y + 1) == '.' and !was.count({x, y + 1})){
    std::vector<char> some_now=gen_algorm(x, y + 1, was);
    if(some_now.size() != 0){
      some_now.push_back('U');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x, y - 1) == '.' and !was.count({x, y - 1})){
    std::vector<char> some_now=gen_algorm(x, y - 1, was);
    if(some_now.size() != 0){
      some_now.push_back('D');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x + 1, y) == '.' and !was.count({x + 1, y})){
    std::vector<char> some_now=gen_algorm(x + 1, y, was);
    if(some_now.size() != 0){
      some_now.push_back('R');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x - 1, y) == '.' and !was.count({x - 1, y})){
    std::vector<char> some_now=gen_algorm(x - 1, y, was);
    if(some_now.size() != 0){
      some_now.push_back('L');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(some.size() != 0)return some;
  if(get(x, y + 1) == 'E'){
    some.push_back('U');
    return some;
  }
  if(get(x, y - 1) == 'E'){
    some.push_back('D');
    return some;
  }
  if(get(x + 1, y) == 'E'){
    some.push_back('R');
    return some;
  }
  if(get(x - 1, y) == 'E'){
    some.push_back('L');
    return some;
  }
  return some;
}

std::pair<int, int> find_robot(){
  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      //std::cout << x << " "<< y << " " << field[num(x, y)] << std::endl;
      if(field[num(x, y)] == '@')return {x, y};
    }
  }
  return {-1, -1};
}


int main(){

  std::cin >> w >> h;

  std::vector<char> fd;

  for(int i =0;i < w*h; i++){
    char now;
    std::cin >> now;
    fd.push_back(now);
  }
  for(int y = h - 1; y >= 0; y--){
    for(int x = 0; x < w; x++){
      field.push_back(fd[num(x, y)]);
    }
  }


  std::pair<int, int> now_map = find_robot();
  //std::cout << now_map.first << " " << now_map.second << std::endl;
  std::set<std::pair<int, int>> was;
  auto commands = gen_algorm(now_map.first, now_map.second, was);

  for(int i = commands.size() - 1; i >= 0; i--){
    std::cout << commands[i];
  }
  std::cout << std::endl;
}
```

--- Solution 48-000006-34206-robo16_031-robo16_031-20160416154659.cpp ---

```
#include <iostream>
#include <vector>
#include <utility>
#include <set>

int h, w;
std::vector<char> field;


int num(int x, int y){
  return x + y*w;
```

```
}

char get(int x, int y){
  if(x < 0 or x >= w)return 'b';
  if(y < 0 or y >= h)return 'b';

  return field[num(x, y)];
}

std::vector<char> gen_algorm(int x, int y, std::set<std::pair<int, int>> was){
  was.insert({x, y});
  std::vector<char> some;
  bool first = true;
  if(get(x, y + 1) == '.' and !was.count({x, y + 1})){
    std::vector<char> some_now=gen_algorm(x, y + 1, was);
    if(some_now.size() != 0){
      some_now.push_back('U');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x, y - 1) == '.' and !was.count({x, y - 1})){
    std::vector<char> some_now=gen_algorm(x, y - 1, was);
    if(some_now.size() != 0){
      some_now.push_back('D');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x + 1, y) == '.' and !was.count({x + 1, y})){
    std::vector<char> some_now=gen_algorm(x + 1, y, was);
    if(some_now.size() != 0){
      some_now.push_back('R');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x - 1, y) == '.' and !was.count({x - 1, y})){
    std::vector<char> some_now=gen_algorm(x - 1, y, was);
    if(some_now.size() != 0){
      some_now.push_back('L');
      if(first or some.size() > some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(some.size() != 0)return some;
  if(get(x, y + 1) == 'E'){
    some.push_back('U');
    return some;
  }
  if(get(x, y - 1) == 'E'){
    some.push_back('D');
    return some;
  }
  if(get(x + 1, y) == 'E'){
    some.push_back('R');
    return some;
  }
  if(get(x - 1, y) == 'E'){
    some.push_back('L');
    return some;
  }
  return some;
}

std::pair<int, int> find_robot(){
  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      //std::cout << x << " "<< y << " " << field[num(x, y)] << std::endl;
      if(field[num(x, y)] == '@')return {x, y};
    }
  }
  return {-1, -1};
}


int main(){

  std::cin >> w >> h;

  std::vector<char> fd;

  for(int i =0;i < w*h; i++){
    char now;
    std::cin >> now;
    fd.push_back(now);
  }
  for(int y = h - 1; y >= 0; y--){
    for(int x = 0; x < w; x++){
      field.push_back(fd[num(x, y)]);
    }
  }


  std::pair<int, int> now_map = find_robot();
  //std::cout << now_map.first << " " << now_map.second << std::endl;
  std::set<std::pair<int, int>> was;
  auto commands = gen_algorm(now_map.first, now_map.second, was);

  for(int i = commands.size() - 1; i >= 0; i--){
    std::cout << commands[i];
  }
  std::cout << std::endl;
}
```

--- Solution 48-000009-34206-robo16_031-robo16_031-20160416161118.cpp ---

```
#include <iostream>
#include <vector>
#include <utility>
#include <set>

int h, w, t;
std::vector<char> field;


int num(int x, int y){
  return x + y*w;
}

char get(int x, int y){
  if(x < 0 or x >= w)return 'b';
  if(y < 0 or y >= h)return 'b';

  return field[num(x, y)];
}
```

```
std::vector<char> gen_algorm(int x, int y, std::set<std::pair<int, int>> was, int counter){
  was.insert({x, y});
  std::vector<char> some;
  bool first = true;
  if(get(x, y + 1) == '.' and !was.count({x, y + 1})){
    std::vector<char> some_now=gen_algorm(x, y + 1, was, counter+1);
    if(some_now.size() != 0){
      some_now.push_back('U');
      if(first or some.size() >= some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x, y - 1) == '.' and !was.count({x, y - 1})){
    std::vector<char> some_now=gen_algorm(x, y - 1, was, counter + 1);
    if(some_now.size() != 0){
      some_now.push_back('D');
      if(first or some.size() >= some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x + 1, y) == '.' and !was.count({x + 1, y})){
    std::vector<char> some_now=gen_algorm(x + 1, y, was, counter + 1);
    if(some_now.size() != 0){
      some_now.push_back('R');
      if(first or some.size() >= some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  if(get(x - 1, y) == '.' and !was.count({x - 1, y})){
    std::vector<char> some_now=gen_algorm(x - 1, y, was, counter + 1);
    if(some_now.size() != 0){
      some_now.push_back('L');
      if(first or some.size() >= some_now.size()){
        first = false;
        some = some_now;
      }
    }
  }
  std::vector<char> new_some;
  if(get(x, y + 1) == 'E'){
    new_some.push_back('U');
    return new_some;
  }
  if(get(x, y - 1) == 'E'){
    new_some.push_back('D');
    return new_some;
  }
  if(get(x + 1, y) == 'E'){
    new_some.push_back('R');
    return new_some;
  }
  if(get(x - 1, y) == 'E'){
    new_some.push_back('L');
    return new_some;
  }
  if(new_some.size() != 0)return new_some;
  return some;
}

std::pair<int, int> find_robot(){
  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      //std::cout << x << " "<< y << " " << field[num(x, y)] << std::endl;
      if(field[num(x, y)] == '@')return {x, y};
    }
  }
  return {-1, -1};
}


int main(){

  std::cin >> w >> h;

  std::vector<char> fd;

  for(int i =0;i < w*h; i++){
    char now;
    std::cin >> now;
    fd.push_back(now);
  }
  for(int y = h - 1; y >= 0; y--){
    for(int x = 0; x < w; x++){
      field.push_back(fd[num(x, y)]);
    }
  }


  std::pair<int, int> now_map = find_robot();
  //std::cout << now_map.first << " " << now_map.second << std::endl;
  std::set<std::pair<int, int>> was;
  auto commands = gen_algorm(now_map.first, now_map.second, was, 1);

  for(int i = commands.size() - 1; i >= 0; i--){
    std::cout << commands[i];
  }
  std::cout << std::endl;
}
```

--- Task B ---

--- Solution 48-000015-34206-robo16_031-robo16_031-20160416165000.cpp ---

```
#include <iostream>
#include <vector>
#include <utility>
#include <set>

int h, w, t;
std::vector<char> field;


int num(int x, int y){
  return x + y*w;
}

char get(int x, int y){
  if(x < 0 or x >= w)return 'b';
  if(y < 0 or y >= h)return 'b';

  return field[num(x, y)];
}
std::vector<char> gen_algorm(int , int, std::set<std::pair<int, int>>, int);

std::vector<char> get_some(int x, int y, std::set<std::pair<int, int>> was, int counter, char to){
  char now = get(x, y);
  if(now == 'E')return {to};
  if((now != '.' and now != 'O' and now != 'C') or was.count({x, y}))return {};
  std::vector<char> some;

  int second_counter = t  - counter % t ;

  if(now == '.') some = gen_algorm(x, y, was, counter + 1);
  else some = gen_algorm(x, y, was, counter + 1 + second_counter);
  if(some.size() == 0)return {};
  some.push_back(to);
  if(now =='O' or now == 'C')for(int i = 0; i < second_counter; i++){some.push_back('S');}


  return some;
}


std::vector<char> gen_algorm(int x, int y, std::set<std::pair<int, int>> was, int counter){
  was.insert({x, y});
  bool first = true;
  std::vector<char> best;
  std::vector<char> now;

  now = get_some(x, y + 1, was, counter, 'U');
  if((first or now.size() < best.size()) and now.size() != 0){
    first = false;
    best = now;
  }
  now = get_some(x, y - 1, was, counter, 'D');
  if((first or now.size() < best.size()) and now.size() != 0){
    first = false;
    best = now;
  }
  now = get_some(x + 1, y, was, counter, 'R');
  if((first or now.size() < best.size()) and now.size() != 0){
    first = false;
    best = now;
  }
  now = get_some(x - 1, y, was, counter, 'L');
  if((first or now.size() < best.size()) and now.size() != 0){
    first = false;
    best = now;
  }
  return best;
}

std::pair<int, int> find_robot(){
  for(int y = 0; y < h; y++){
    for(int x = 0; x < w; x++){
      //std::cout << x << " "<< y << " " << field[num(x, y)] << std::endl;
      if(field[num(x, y)] == '@')return {x, y};
    }
  }
  return {-1, -1};
}


int main(){

  std::cin >> w >> h >> t;

  std::vector<char> fd;

  for(int i =0;i < w*h; i++){
    char now;
    std::cin >> now;
    fd.push_back(now);
  }
  for(int y = h - 1; y >= 0; y--){
    for(int x = 0; x < w; x++){
      field.push_back(fd[num(x, y)]);
    }
  }


  std::pair<int, int> now_map = find_robot();
  //std::cout << now_map.first << " " << now_map.second << std::endl;
  std::set<std::pair<int, int>> was;
  auto commands = gen_algorm(now_map.first, now_map.second, was, 0);

  for(int i = commands.size() - 1; i >= 0; i--){
    std::cout << commands[i];
  }
  std::cout << std::endl;
}
```