KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:

IFF-1/1 gr. studentas
Vytenis Kriščiūnas
2023 m. vasario 22 d.

Priėmė:

Doc. Dr. Svajūnas Sajavičius

KAUNAS 2023

TURINYS

1. C++20 arba Ruby (L1)

1.1. *Darbo užduotis*

278 Chess

Almost everyone knows the problem of putting eight queens on an 8×8 chessboard such that no Queen can take another Queen. Jan Timman (a famous Dutch chessplayer) wants to know the maximum number of chesspieces of one kind which can be put on an m × n board with a certain size such that no piece can take another. Because it's rather difficult to find a solution by hand, he asks your help to solve the problem.

He doesn't need to know the answer for every piece. Pawns seems rather uninteresting and he doesn't like Bishops anyway. He only wants to know how many Rooks, Knights, Queens or Kings can be placed on one board, such that one piece can't take any other.

Input

The first line of input contains the number of problems. A problem is stated on one line and consists of one character from the following set 'r', 'k', 'Q', 'K', meaning respectively the chesspieces Rook, Knight, Queen or King. The character is followed by the integers m ($4 \le m \le 10$) and n ($4 \le n \le 10$), meaning the number of rows and the number of columns or the board.

Output

For each problem specification in the input your program should output the maximum number of chesspieces which can be put on a board with the given formats so they are not in position to take any other piece.

Note: The bottom left square is 1, 1.

Sample Input

```
2
r 6 7
k 8 8
```

Sample Output

```
6
32
```

1.2. *Programos tekstas*

```
#include <fstream>
#include <iostream>
#include <format>
#include <chrono>
#include <list>
using namespace std;

class Piece
{
public:
	char Name;
	int M, N;

	Piece(char name, int m, int n) : Name(name), M(m), N(n) {}
};

class Pieces
{
private:
	list<Piece> pieces;
	int Number;
public:
	Pieces(int number = 0) : Number(number){}

	void Add(Piece piece)
	{
		pieces.push_back(piece);
	}

	Piece Get(int index)
	{
		list<Piece>::iterator it = pieces.begin();
		advance(it, index);
		return *it;
	}

	int GetNumber()
	{
		return Number;
	}
};

class InOutUtils
{
public:
	static Pieces Read(const string fileName)
	{
		int number, m, n;
		char name;

		ifstream file(fileName);
		if (!file.is_open())
		{
			cerr << "Unable to open a file " << fileName << endl;
			return 0;
		}

		file >> number;
		Pieces pieces = Pieces(number);
		for (int i = 0; i < number; i++)
		{
			file >> name >> m >> n;
			Piece piece = Piece(name, m, n);
			pieces.Add(piece);
		}
		file.close();
		return pieces;
	}

	static void Write(const string fileName, list<int> totalCounts)
	{
		ofstream file(fileName);
		if (!file.is_open())
		{
			cerr << "Unable to open a file " << fileName << endl;
			return;
		}

		for (int count : totalCounts)
		{
			file << count << endl;

		}
		file.close();
	}
};
```

```
class TaskUtils
{
public:
	static list<int> Calculate(Pieces pieces)
	{
		list<int> totalCounts;

		for (int i = 0; i < pieces.GetNumber(); i++)
		{
			Piece piece = pieces.Get(i);
			if (piece.Name == 'r' || piece.Name == 'Q')
			{
				totalCounts.push_back(piece.M);
			}
			else if (piece.Name == 'K')
			{
				int m = piece.M / 2 + piece.M % 2;
				int n = piece.N / 2 + piece.N % 2;
				totalCounts.push_back(m * n);
			}
			else
			{
				int m1 = piece.M / 2 + piece.M % 2;
				int m2 = piece.M / 2;
				int n1 = piece.N / 2 + piece.N % 2;
				int n2 = piece.N / 2;
				int rezult = m1 * n1 + m2 * n2;
				totalCounts.push_back(rezult);
			}

		}
		return totalCounts;
	}
};

int main()
{
	typedef chrono::high_resolution_clock Time;
	typedef chrono::duration<float> duration;

	auto start = Time::now();

	Pieces pieces = InOutUtils::Read("inputFile.txt");

	list<int> totalCounts = TaskUtils::Calculate(pieces);

	InOutUtils::Write("Rezults.txt", totalCounts);

	auto stop = Time::now();

	duration totalTime = chrono::duration_cast<chrono::microseconds>(stop - 
start);
	cout << "Duration: " << totalTime.count() << " ms." <<endl;
	return 0;
	}
```

1.3. Pradiniai duomenys ir rezultatai

inputFile.txt tekstas:

```
24
K 8 10
K 7 9
K 7 7
K 8 8
```

```
K 6 10
K 5 10

Q 8 10
Q 7 9
Q 7 7
Q 8 8
Q 6 10
Q 5 10

k 8 10
k 7 9
k 7 7
k 8 8
k 6 10
k 5 10


r 8 10
r 7 9
r 7 7
r 8 8
r 6 10
r 5 10
```

Rezults.txt tekstas:

```
20
20
16
16
15
15
8
7
7
8
6
5
40
32
25
32
30
25
8
7
7
8
6
5
```

Konsolės rezultatai:

```
Duration: 0.002962 ms.
```

2. **Scala (L2)**
3. **Haskell (L3)**
4. **Prolog (L4)**