Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
Estrutura de Dados I

Gotta Catch'em All

Especificações do Trabalho de Fundamentos de Programação I

Sumário

1 Datas importantes

02/12/2019	Entrega do trabalho (S01 e S02)
03/12/2019	Atividade Prática Supervisionada (S02)
04/12/2019	Atividade Prática Supervisionada (S01)
09/12/2019	Divulgação da correção (S01 e S02)
16/12/2019	Re-entrega do trabalho (S01 e S02)

2 Enunciado do problema

Você é um renomado treinador de *Pokémon* que está num bosque junto com alguns outros treinadores para jogar um jogo. No bosque há vários *Pokémon* selvagens prontos para serem capturados. Você possui uma quantidade infinita de *Pokéballs*, porém uma quantidade finita de pontos de energia. Cada captura de um *Pokémon* custa alguns pontos de sua energia. Após acabar sua energia, você estará cansado demais e não conseguirá mais caminhar. Seu objetivo no jogo é capturar mais *Pokémon* que os outros treinadores, independentemente de quanta energia cada *Pokémon* consuma para ser capturado.

Cada partida do jogo é definida por uma sequência de rodadas. Em cada rodada, cada treinador decide como quer se movimentar no bosque. Se algum treinador se mover sobre uma posição na qual se encontra um *Pokémon*, e se o treinador possuir energia suficiente para capturar aquele *Pokémon*, então aquele *Pokémon* será capturado por aquele treinador.

Uma partida dá-se por encerrada quando não há mais *Pokémon* a serem capturados ou quando numa rodada ocorre de nenhum treinador se mover.

3 Submissão e avaliação

Este trabalho deverá ser desenvolvido por equipes de até 3 integrantes. A composição das equipes é livre e só é identificada na ocasião da submissão dos trabalhos.

Cada equipe deverá fazer a submissão do trabalho pelo Moodle através da conta de apenas um dos integrantes da equipe. A submissão deverá consistir num único arquivo `.c` contendo unicamente a implementação da função

```
int pokemon(char a[100][100], char you, int energy);
```

a qual, conforme especificado na Seção 4, recebe a descrição de uma configuração do jogo e devolve qual é o próximo movimento no bosque que você decide fazer. Assuma que chamadas sucessivas a esta função, bem como a outras funções similares, serão realizadas por uma função `main()` implementada pelo professor, a fim de executar uma partida do jogo para possivelmente vários treinadores de *Pokémon*.

O arquivo submetido deve ser nomeado de acordo com a concatenação dos nomes completos de todos os integrantes da equipe, separando-se diferentes integrantes por um hífen, e removendo-se dos nomes todos os espaços e diacríticos. Por exemplo, se integram uma equipe os estudantes Ciça Guimarães, John Lennon, e Luis Silva, o nome do arquivo deverá ser

`CicaGuimaraes-JohnLennon-LuisSilva.c`

Todos os integrantes de uma mesma equipe receberão a mesma nota

$$T = \max \begin{cases} T' \\ 0{,}4T' + 0{,}6T^R \end{cases}$$

pelo trabalho, sendo T' a nota atribuída à primeira entrega e T^R a nota atribuída à re-entrega. Não serão permitidas mudanças na composição das equipes entre a entrega e a re-entrega.

Se houver algum estudante que seja identificado como pertencente a mais de uma equipe, a nota T desse estudante será anulada. Se alguma equipe realizar uma submissão com mais de 3 integrantes, a nota T de todos os integrantes dessa equipe será anulada.

É permitido que as equipes copiem os códigos desenvolvidos pelo professor livremente. A troca de dicas e ideias entre as equipes também é encorajada. No entanto, trechos de códigos que sejam constrangedoramente semelhantes a códigos de outros autores que não o professor poderão configurar plágio e acarretar na anulação da nota T de todos os integrantes da equipe.

Os trabalhos serão avaliados sob estes critérios:

- compilação;
- corretude;

- eficiência de tempo;
- elegância do código;
- competitividade em relação aos demais trabalhos entregues.

4 Especificações técnicas

Implemente apenas uma função de cabeçalho

```
char pokemon(int n, int m, char a[][100], char you, int energy);
```

Esta função recebe:

- dois inteiros positivos n e m, ambos não maiores que 100;
- um *array* bidimensional a[][], de dimensões $n \times m$, de caracteres, o qual representa o bosque;
- o caractere *you*, uma letra minúscula do alfabeto que representa como é identificada em a[][] a posição onde você se encontra;
- o número de pontos de energia que ainda lhe resta.

Assuma que cada caractere a$[i][j]$ em posição válida — ou seja, cada caractere a$[i][j]$ que satisfaz $0 \leq i < n$ e $0 \leq j < m$ — pode cair em apenas uma das possibilidades listadas a seguir:

- `#`, indicando a presença de uma árvore do bosque naquela posição (neste caso, a posição é intransponível e não contém *Pokémon* algum);
- um algarismo x ($0 \leq x \leq 9$), indicando que naquela posição há um *Pokémon* que consome x pontos de energia para ser capturado;
- uma letra minúscula do alfabeto, indicando que naquela posição há um treinador, representado por aquela letra, sendo que dois treinadores não podem ocupar a mesma posição do bosque;
- um espaço em branco, indicando que nada há naquela posição.

Assuma também que todos os parâmetros passados para a função são válidos. Em particular, assuma que exatamente um dos caracteres é igual ao caractere *you* e que há ainda ao menos um *Pokémon* a ser capturado. Porém, **não** é garantido que em alguma das posições inválidas o *array* a[][] contém o caractere '`\0`' (i.e. as linhas do *array* a[][] não devem ser tratadas como *strings*).

A função `pokemon()` deve retornar um dentre os seguintes caracteres:

- `N` (maiúsculo), indicando que você está decidindo se mover em uma posição na direção norte (i.e. se você está em a$[i][j]$, agora você deseja ir para a$[i-1][j]$);
- `S` (maiúsculo), indicando que você está decidindo se mover em uma posição na direção sul (i.e. se você está em a$[i][j]$, agora você deseja ir para a$[i+1][j]$);

- `E` (maiúsculo), indicando que você está decidindo se mover em uma posição na direção leste (i.e. se você está em a$[i][j]$, agora você deseja ir para a$[i][j+1]$);
- `W` (maiúsculo), indicando que você está decidindo se mover em uma posição na direção oeste (i.e. se você está em a$[i][j]$, agora você deseja ir para a$[i][j-1]$);
- `X` (maiúsculo), indicando que você está decidindo permanecer na sua posição corrente.

Caso a função `pokemon()` retorne um movimento não considerado permitido, você continuará na sua posição corrente. Em particular,

- não é permitido que você decida se mover para uma posição para fora dos domínios do bosque;
- não é permitido que você decida se mover para uma posição onde se encontra uma árvore;
- não é permitido que você decida se mover para uma posição na qual se encontra um *Pokémon* para o qual você não tenha pontos de energia suficiente;
- não é permitido que você saia da posição na qual você se encontra se você possui zero pontos de energia;
- não é permitido que você decida se mover para uma posição para a qual também deseja ir outro treinador, situação em que nenhum dos treinadores acabará saindo do lugar.

Para a implementação da função `pokemon()` pode ser assumida a inclusão das bibliotecas `stdio.h`, `stdlib.h`, `string.h`, `time.h`, `math.h`, e apenas essas. A compilação do código-fonte que vai incluir a função `pokemon()` deve também ser assumida sob exatamente este conjunto de *flags* de compilação para o `gcc`:

```
-Wall -O2 -ansi -Wno-unused-result -lm
```

A compilação da função `pokemon()` deve ocorrer sem acusar erro ou aviso (*warning*) algum.

5 Exemplo

Suponhamos um jogo entre dois treinadores `a` e `b`, sendo você o treinador `a`, e `b` o seu oponente. Para o *array* 3×10 a seguir, por exemplo, numa situação em que você possui 10 pontos de energia,

```
12300#####
 9a91 b   1
  ## 999 2
```

é permitido qualquer um dentre os movimentos `N`, `W`, e `E`, mas não é permitido o movimento `S` (observe que a opção `X` de não se movimentar sempre é permitida). Se sob a mesma configuração do jogo você possuísse apenas 8 pontos de energia, apenas o movimento `N` seria permitido (além da opção `X` do não-movimento). Agora, se porventura a situação fosse

```
1###0#####
 #a0b   1 2
  ## 999 2
```

sua função pode retornar apenas `E` ou `X`, independentemente do número de pontos de energia, mas ainda assim a decisão `E` poderia não ser validada, caso o treinador `b` tomasse a decisão `W`.