MAC2166 – Introdução à Computação

Escola Politécnica

Segunda Prova – 19 de maio de 2015

Nome: _____

Assinatura: _____

Nº USP: _____ Turma: _____

Professor: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. A prova consta de 2 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. Qualquer questão pode ser resolvida em qualquer página. Se a questão não está na página correspondente ao enunciado basta indicar isto na página e escrever QUESTÃO X em letras ENORMES antes da solução.
4. A prova pode ser feita a lápis. Cuidado com a legibilidade e, principalmente, com a TABULAÇÃO.
5. Não é necessário apagar rascunhos no caderno de questões.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é permitido o uso de equipamentos eletrônicos.
8. Não é permitido a consulta a livros, apontamentos ou colegas.
9. Você pode definir funções e usá-las à vontade.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1(a)	2,5	
1(b)	1,0	
1(c)	3,5	
2	3,0	
Total	10,0	

QUESTÃO 1 (vale 7,0 pontos)

Nessa questão, considere uma versão simplificada do jogo Sokoban. As constantes definidas abaixo devem ser OBRIGATORIAMENTE utilizadas nessa questão, sem precisar redefini-las. Observe que há apenas 4 elementos possíveis em um mapa simplificado, e também apenas 4 movimentos possíveis.

```
# elementos possíveis em um mapa
CAIXA   = '$'
JOGADOR = '@'
PAREDE  = '#'
VAZIO   = ' '  # espaço indicando piso vazio

# movimentos possíveis
BAIXO   = 'b'
CIMA    = 'c'
DIR     = 'd'
ESQ     = 'e'
```

Considere também as seguintes quatro funções:

```
def  carregaMapa():
    """(None) -> mapa
       Retorna um mapa com uma
       configuração simplificada
       de Sokoban.
       Um mapa é uma lista de listas de
       strings (caracteres) como no EP,
       são cercados por paredes e sempre
       possuem um único jogador. Não é
       necessário verificar se o mapa é
       válido.
    """


def achaJogador(mapa):
    """(mapa) -> int, int
       Retorna a posição (lin,col) do
       jogador ou None caso não existir
       jogador no mapa
    """
```

```
def imprimeMapa(mapa):
    """(mapa) -> None
       Imprime um mapa com moldura
    """


def moveJogador(mapa, mov):
    """(mapa, str) -> bool, bool, int, int
        Recebe um mapa e um movimento mov.
        Se mov é válido, a função atualiza mapa
        e retorna os seguintes 4 valores:
        - 1) True indicando que mov é válido
        - 2) True/False indicando se uma
             caixa foi empurrada
        - 3) nova linha do jogador.
        - 4) nova coluna do jogador.

        Caso o movimento seja inválido,
        retorna False, False, -1, -1.
    """
```

Nos 3 itens dessa questão, você deve escrever um programa (função `main`) e as funções `achaJogador` e `moveJogador`. Não é permitido modificar o protótipo dessas funções.

O programa deve simular os movimentos do jogador de acordo com uma sequência de movimentos dada. Como no EP, em um movimento válido, o jogador pode se mover para posições vazias ou empurrar uma caixa para uma posição vazia. Movimentos inválidos são aqueles que tentam mover o jogador para uma posição com parede ou com uma caixa presa, ou seja, encostada na parede ou encostada em outra caixa.

Item a (vale 2,5 pontos)

Escreva um programa (função `main`) que carrega um `mapa`, lê uma sequência `movs` de movimentos na forma de um único string, realiza os movimentos válidos atualizando o `mapa` e enviando mensagens **exatamente** como no exemplo abaixo. Ao final o programa deve imprimir as jogadas válidas realizadas, sendo que os movimentos que resultaram em empurrões de caixas devem ser impressos em letras maiúsculas, como no EP.

O string com os movimentos pode conter qualquer sequência dos seguintes caracteres: 'b', 'c', 'd', 'e' ('b' = baixo, 'c' = cima, 'd' = direita, 'e' = esquerda). Um movimento no string só deve ser realizado se for válido. Movimentos inválidos devem ser ignorados, e o processo deve continuar processando os movimentos seguintes, até o final do string.

O seu programa deve utilizar, obrigatoriamente, todas as funções descritas anteriormente, a saber: `carregaMapa`, `imprimeMapa`, `achaJogador` e `moveJogador`, **sem escrevê-las** (a `achaJogador` será escrita no item b e a `moveJogador` será escrita no item c).

Exemplo de execução para a sequência de comandos (string) "bbdc"(que é digitada pelo usuário, após o programa exibir a configuração inicial). Lembre-se que o seu programa deve imprimir mensagens **exatamente** como nesse exemplo.

```
Configuração inicial:
      0   1   2   3   4   5   6
    +---+---+---+---+---+---+---+
  0 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
  1 |   | # |   | @ |   | # |   |
    +---+---+---+---+---+---+---+
  2 | # |   |   | $ |   |   | # |
    +---+---+---+---+---+---+---+
  3 | # |   |   |   |   | $ | # |
    +---+---+---+---+---+---+---+
  4 | # |   |   | $ |   |   | # |
    +---+---+---+---+---+---+---+
  5 |   | # |   |   |   | # |   |
    +---+---+---+---+---+---+---+
  6 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
Posição inicial: 1, 3

Digite seus movimentos: bbdc

>>>> Movimento b válido

      0   1   2   3   4   5   6
    +---+---+---+---+---+---+---+
  0 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
  1 |   | # |   |   |   | # |   |
    +---+---+---+---+---+---+---+
  2 | # |   |   | @ |   |   | # |
    +---+---+---+---+---+---+---+
  3 | # |   |   | $ |   | $ | # |
    +---+---+---+---+---+---+---+
  4 | # |   |   | $ |   |   | # |
    +---+---+---+---+---+---+---+
  5 |   | # |   |   |   | # |   |
    +---+---+---+---+---+---+---+
  6 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
O jogador moveu uma caixa e
O jogador está na posição: 2, 3

>>>> Movimento b inválido

>>>> Movimento d válido

      0   1   2   3   4   5   6
    +---+---+---+---+---+---+---+
  0 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
  1 |   | # |   |   |   | # |   |
    +---+---+---+---+---+---+---+
  2 | # |   |   |   | @ |   | # |
    +---+---+---+---+---+---+---+
  3 | # |   |   | $ |   | $ | # |
    +---+---+---+---+---+---+---+
  4 | # |   |   | $ |   |   | # |
    +---+---+---+---+---+---+---+
  5 |   | # |   |   |   | # |   |
    +---+---+---+---+---+---+---+
  6 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
O jogador está na posição: 2, 4

>>>> Movimento c válido

      0   1   2   3   4   5   6
    +---+---+---+---+---+---+---+
  0 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
  1 |   | # |   |   | @ | # |   |
    +---+---+---+---+---+---+---+
  2 | # |   |   |   |   |   | # |
    +---+---+---+---+---+---+---+
  3 | # |   |   | $ |   | $ | # |
    +---+---+---+---+---+---+---+
  4 | # |   |   | $ |   |   | # |
    +---+---+---+---+---+---+---+
  5 |   | # |   |   |   | # |   |
    +---+---+---+---+---+---+---+
  6 |   |   | # | # | # |   |   |
    +---+---+---+---+---+---+---+
O jogador está na posição: 1, 4

Os movimentos válidos foram:
Bdc
```

```
def main():
```

Item b (vale 1,0 pontos) Escreva a função `achaJogador`.

```
def achaJogador(mapa):
    """(mapa) -> int, int"""
```

Item c (vale 3,5 pontos) Escreva a função `moveJogador`. Você pode usar a função `achaJogador`, sem escrevê-la, mesmo que não a tenha feito.

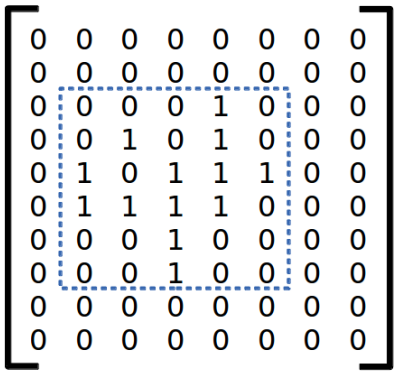
```
def moveJogador(mapa, mov):
    """(mapa, str) -> bool, bool, int, int"""
```

QUESTÃO 2 (vale 3,0 pontos)

Escreva uma função em Python que dada uma matriz de inteiros binária (isto é, contendo valores 0 ou 1) devolve uma segunda matriz correspondendo ao menor recorte retangular da primeira, que contém todos elementos com valores iguais a 1 (matriz delimitadora mínima).

OBS: Assuma que a matriz sempre tem pelo menos um elemento com valor 1.

Exemplo: Para a matriz de entrada:



A nova matriz criada e devolvida pela função será: