Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Estruturas de Dados I – BCC202
Professor: David Menotti (menottid@gmail.com)
Professora: Emiliana Mara Lopes Simões (simoes.eml@gmail.com)

Trabalho Prático 2 – Dominó [1]

Listas x Conjuntos

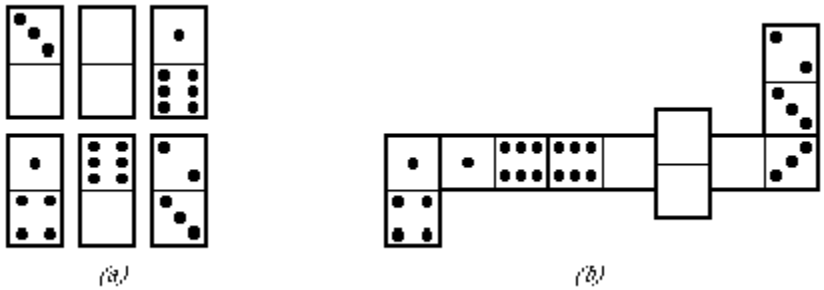
Valor: 0,5 pontos (5% da nota total) **Data de entrega: 26/04/2010**
Documentação em Latex: 0,1 pontos extra
Interface gráfica: até 0,2 pontos extra
Impressão frente-verso: 0,05 pontos extra

Objetivos

Consiste em utilizar conceitos de listas para solução de um jogo bem conhecido.

Descrição

Todos conhecem o jogo de dominós, em que peças com dois valores devem ser colocadas na mesa em seqüência, de tal forma que os valores de peças imediatamente vizinhas sejam iguais. O objetivo desta tarefa é determinar se é possível colocar todas as peças de um dado conjunto em uma formação válida. Veja um exemplo ilustrativo abaixo.



Conjunto de seis peças (a) e uma formação utilizando todas as seis peças (b)

Tarefa

É dado um conjunto de peças de dominó. Cada peça tem dois valores X e Y, com X e Y variando de 0 a 6 (X pode ser igual a Y). Sua tarefa é escrever um programa que determine se é possível organizar todas as peças recebidas em sequência, obedecendo às regras do jogo de dominó.

Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de testes contém um número inteiro N que indica a quantidade de peças do conjunto. As N linhas seguintes contêm, cada uma, a descrição de uma peça. Uma peça é descrita por dois inteiros X e Y ($0 \leq X \leq 6$ e $0 \leq Y \leq 6$) que representam os valores de cada lado da peça. O final da entrada é indicado por N = 0.

Exemplo de Entrada

```
3
0 1
2 1
2 1
2
1 1
0 0
6
3 0
0 0
1 6
4 1
0 6
2 3
0
```

Saída

Para cada conjunto de teste da entrada seu programa deve produzir quatro linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste n", onde n é numerado a partir de 1. A segunda linha deve conter a expressão “sim” se for possível organizar todas as peças em uma formação válida ou a expressão “nao” (note a ausência de acento) caso contrário. A terceira linha deve apresentar a sequência de alocação das peças (separadas por “|”, caso seja possível organizá-las ou “não” caso contrário. A quarta linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
sim
01|12|21

Teste 2
nao
nao

Teste 3
sim
23|30|00|06|61|14
```

(esta saída corresponde ao exemplo de entrada acima)

Restrições

$0 \leq N \leq 28$ ($N = 0$ apenas para indicar o final da entrada)

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Estruturas de Dados I – BCC202
Professor: David Menotti (menottid@gmail.com)
Professora: Emiliana Mara Lopes Simões (simoes.eml@gmail.com)

O que deve ser entregue

- Código fonte do programa em C ou C++ (bem identado e comentado).
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:
  1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. **Muito importante**: os códigos utilizados nas implementações devem ser inseridos na documentação.
  3. Análise de Complexidade: estudo da complexidade de tempo e espaço das funções implementadas e do programa como um todo (notação O).
  4. Listagem de testes executados: os testes executados devem ser apresentados e analisados e discutidos, quando convier.
  5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto quando da sua utilização
  7. Em Latex: Caso o trabalho seja elaborado/escrito em latex, ganha-se 0,1 pontos.
  8. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

  Obs: Veja modelo de como fazer o trabalho em latex:
  http://www.decom.ufop.br/menotti/edI101/tps/modelo.zip

Como deve ser feita a entrega:

A entrega DEVE ser feita via Moodle (www.decom.ufop.br/moodle) na forma de um **único** arquivo zipado, contendo o código, os arquivos e a documentação. Também deve ser entregue a documentação impressa na próxima aula teórica após a data de entrega do trabalho.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, identação e comentários no programa também serão avaliados;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero. Devido a recorrentes problemas com cópias de trabalhos (plágios), os autores de trabalhos copiados também terão a maior nota nos testes teóricos levada a zero, como forma de punição e coação ao plágio acadêmico;
- Trabalhos entregues em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.

Referências

[1] Olímpiadas de Programação, http://olimpiada.ic.unicamp.br/ (sítio de internet), visitado em 12/04/2010.