UNIVERSIDADE FEDERAL DE MATO GROSSO
CAMPUS UNIVERSITÁRIO DO ARAGUAIA
Instituto de Ciências Exatas e da Terra
Curso de Bacharelado em Ciência da Computação

Disciplina: Estrutura de Dados I
Professor: Ivairton M. Santos
Ano: 2017/2.

Trabalho – Listas com Alocação Dinâmica e Hash

Problema 1 – A polícia da cidade *"Tá brabo o negócio"* resolveu desenvolver um sistema para cadastrar os suspeitos da região. A polícia pretende cadastrar os suspeitos considerando o grau de periculosidade, dessa forma, os primeiros da lista são os mais perigosos. Os dados a serem armazenados são o nome e a idade do suspeito.

Considerando a estrutura de dados **lista duplamente encadeada**, defina o tipo de dado abstrado a ser utilizado e implemente um sistema que tenha as seguintes funcionalidades:

- Registrar um meliante, sem especificação de periculosidade (portanto, ao final da lista com grau mínimo);
- Registrar um meliante em posição específica da lista (classificando portanto, quando à periculosidade em relação àqueles já cadastrados);
- Remover um meliante através do nome;
- Remover um meliante através da posição na lista;
- Listar todos os meliantes presentes na lista (de acordo com a periculosidade);
- Listar todos os meliantes presentes na lista de acordo com a idade (os mais velhos antes dos mais jovens);

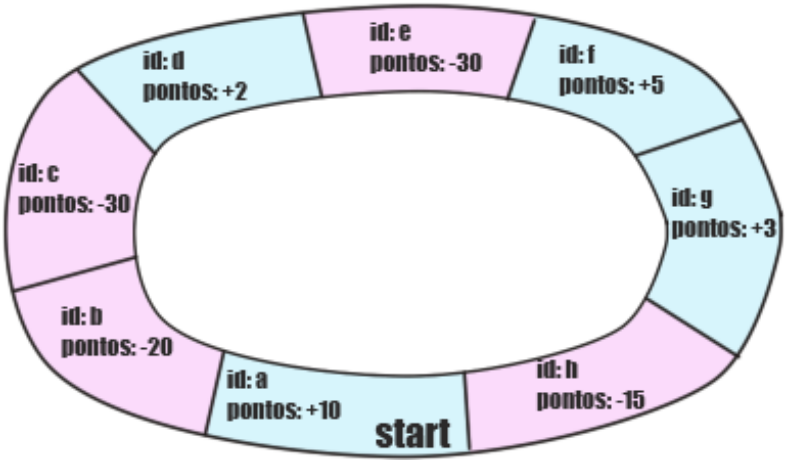
Problema 2 – Implemente uma lista circular (com encadeamento simples) que registre um conjunto de números inteiros positivos a partir de um arquivo de entrada do tipo texto. Com a lista preenchida, implemente uma função que imprima todos os números presentes na lista circular de trás para frente, percorrendo a lista somente uma vez.

Problema 3 – GoM é um jogo de tabuleiro com estrutura descrita e tabuleiro ilustrados abaixo:

Cada posição no tabuleiro contém um identificador (id) e uma quantidade de pontos, que pode ser positivo ou negativo. Todos os jogadores iniciam o jogo na primeira posição (demarcada com "*Start*") e a pontuação inicial para todos é o valor absoluto da posição (no exemplo, abs(+10)).

O jogo consiste em cada jogador rolar um dado e avançar a quantidade de posições equivalentes ao valor do obtido com a jogada. Ele então e ganha, ou perde, a quantidade de pontos marcada na posição do tabuleiro alcançada.

Caso o jogador caia em uma posição com pontuação negativa, na sua próxima jogada irá retroceder (sentido anti-horário) o número de posições obtidas com o jogo do dado. Caso um jogador caia em uma casa com pontuação positiva, na sua próxima jogada avançará as posições no sentido horário.



Caso um jogador fique com sua quantidade total de pontos negativa ele perde e é removido do jogo. O jogo acaba quando sobrar apenas um jogador.

Faça um programa que simule este jogo!

A entrada deve ser separada em três partes:

- **Entrada para o tabuleiro:** Um inteiro “T” que indica a quantidade de posições no tabuleiro. A seguir, T caracteres “id” (tal que, ‘a’ <= id <= ‘z’) que representarão o nome da posição; e inteiros “pontos” que representará a pontuação atribuída para aquela posição (pode ser positivo ou negativo);
- **Entrada dos jogadores:** Um inteiro “N” representando a quantidade de jogadores. A seguir, N strings “nome” representando o nome de cada jogador;
- **Entrada das jogadas:** As jogadas são compostas por vários “Numero_tirado_no_Dado” (que representa a quantidade de casas que cada jogador deve se movimentar). A primeira jogada é referente ao primeiro jogador adicionado na “Entrada dos jogadores”, a segunda jogada é referente ao segundo jogador e assim sucessivamente. Novas jogadas devem ser lidas até que sobre apenas um jogador.

O programa deve imprimir o nome do jogador vencedor e sua quantidade total de pontos.

Observação: Utilize uma **lista duplamente encadeada** circular para representar o tabuleiro e uma **lista circular** para representar os jogadores.

Exemplo do arquivo de entrada para o tabuleiro:

```
8
a      10
b      -20
c      -30
d      2
e      -30
f      5
g      3
h      -15
```

Exemplo de entrada para os jogadores:

```
6
Red
Yellow
Purple
Blue
Green
Black
```

Exemplo de entrada para as jogadas do dado:

```
3
1
2
5
4
7
2
1
3
1
3
3
```

Simulação do exemplo dado:

- Jogador 1 (Red) avança 3 posições no sentido horário, cai na casa “d” e ganha 2 pontos. (12 total)
- Jogador 2 (Yellow) avança 1 posição no sentido horário, cai na casa “b” e perde 20 pontos, sua pontuação fica negativa e ele é removido do jogo.
- Jogador 3 (Purple) avança 2 posições no sentido horário, cai na casa “c” e perde 30 pontos, sua pontuação fica negativa e ele é removido do jogo.
- Jogador 4 (Blue) avança 5 posições no sentido horário, cai na casa “f” e ganha 5 pontos. (15 total)
- Jogador 5 (Green) avança 4 posições no sentido horário, cai na casa “e” e perde 30 pontos, sua pontuação fica negativa e ele é removido do jogo.
- Jogador 6 (Black) avança 7 posições no sentido horário, cai na casa “h” e perde 15 pontos, sua pontuação fica negativa e ele é removido do jogo.
- Jogador 1 (Red) avança 2 posições no sentido horário, cai na casa “f” e ganha 5 pontos. (17 total)
- Jogador 4 (Blue) avança 1 posição no sentido horário, cai na casa “g” e ganha 3 pontos. (18 total)
- Jogador 1 (Red) avança 3 posições no sentido horário, cai na casa “a” e ganha 10 pontos. (27 total)
- Jogador 4 (Blue) avança 1 posição no sentido horário, cai na casa “h” e perde 15 pontos. (3 total). Na sua próxima jogada ele avançará em sentindo anti-horário.
- Jogador 1 (Red) avança 3 posições no sentido horário, cai na casa “d” e ganha 2 pontos. (29 total)
- Jogador 4 (Blue) avança 3 posições no sentido anti-horário, cai na casa “e” e perde 30 pontos, sua pontuação fica negativa e ele é removido do jogo.

Saída:

Vencedor: Red (29 pontos).

Problema 4 - Implemente um dicionário onde cada entrada é dada da forma:

`<palavra> <definição>.`

Como dicionários costumam ter um volume muito grande de dados, utilizar uma lista normal para esta tarefa pode ser inviável.

Implemente uma Tabela Hash (com tratamento de colisão por lista ligada) para armazenar as palavras deste dicionário.

a) Defina a função de dispersão. Como palavras podem ser traduzidas para um índice da tabela? (Dica: Utilize a tabela ASCII).
b) Faça uma função para inserir novas palavras no dicionário.
c) Faça uma função que leia uma palavra e devolva a definição caso ela esteja presente no dicionário.

Problema 5 - Implemente uma lista telefônica que funciona da seguinte forma:

***i*)** Cada código postal tem vários números de telefone com o código de área correspondente a este código postal.
***ii*)** Cada número de telefone tem o nome do proprietário da linha.

Represente esta lista como uma Tabela Hash:

Cada **código postal** deverá ter:

***i*)** a sua chave única na Tabela Hash;
***ii*)** uma outra Tabela Hash que representa todos os números de telefone na região deste código postal.

Cada **número de telefone** deverá ter:

i) a sua chave única na Tabela Hash;
ii) o nome do seu proprietário.

Implemente uma função para cada uma das funcionalidades a seguir:
***a*)** Inserção de novos códigos postais.
***b*)** Inserção de novos números de telefone+nome do proprietário com a informação de código postal válido.
***c*)** Busca/impressão de todos os números de telefone em um determinado código postal.
***d*)** Busca/impressão do nome do proprietário dado um código postal e um número de telefone.