Instituto Federal de Educação, Ciência e Tecnologia

Câmpus Campinas
ED1 – Estruturas de Dados I

Professores: José Américo (jose.americo@ifsp.edu.br)
Samuel Martins (samuel.martins@ifsp.edu.br)

Lab 05 – Playlist Musical

1) Descrição

Após um semestre árduo, Patrícia decidiu aproveitar suas férias para viajar. Como ela não consegue dormir durante as viagens, ela planeja usar esse tempo pra ouvir músicas em seu celular (com fone de ouvido, claro!).

Patrícia usa a plataforma IspotiFaia para ouvir suas belas canções. Ela então criou várias playlists (lista de músicas) para ouvir na viagem. O player da plataforma possui um indicador que mostra o nome da música na posição atual da playlist. Caso a playlist esteja vazia, como é o caso do ínicio da execução (criação da playlist), o indicador exibirá a string vazia.

O player possui as seguintes operações sob uma playlist:

- **insere NOME_DA_MUSICA**
  - insere a música de nome NOME_DA_MUSICA na playlist, logo após a posição atual do cursor;
  - Caso a playlist esteja vazia, a música é inserida na primeira posição;
  - Uma música pode aparecer na playlist mais de uma vez.
- **remove NOME_DA_MUSICA**
  - Remove da playlist **a primeira ocorrência** da música de nome NOME_DA_MUSICA;
  - Se a música removida for a música no indicador/cursor, este deve ser deslocado, caso haja mais músicas na playlist. **Ele sempre é deslocado para a música anterior**, a não ser que a música removida tenha sido a primeira da lista. Neste caso, o indicador exibirá a música que agora é a primeira.
  - Caso a música não conste na playlist ou a playlist esteja vazia, nada acontece.
- **toca**
  - toca a música mostrada pelo indicador;
  - Após o fim da execução, o indicador passa para a próxima música;
  - Se a música tocada for a última da playlist, o indicador passará a ser a primeira música da playlist;
  - Se a playlist estiver vazia, o indicador exibirá a string vazia (**printf("\n");**).

- **volta**
    - Volta o indicador uma posição na playlist;
    - Caso o indicador esteja na primeira música, ele vai para a última música;
    - Se a playlist estiver vazia, nada acontece.

- **final**
    - Começando da posição atual do indicador, esta instrução executa a música atual e todas as músicas seguintes, até o fim da playlist, **deixando o indicador na última música;**
    - Se a playlist estiver vazia, o indicador exibirá a string vazia (**printf("\n");**);
- **inverte**
    - inverte a ordem de todas as músicas na playlist;
    - O cursor continuará apontando para a mesma música (nó) do indicador após a inversão;
    - Se a playlist estiver vazia, nada acontece.

Caso alguma das operações que envolvem deslocamento da playlist seja executada quando a lista estiver vazia, não há deslocamento e o tocador não retorna nenhum erro.

Sua tarefa será desenvolver o tocador da plataforma IspotiFaia, que vai imprimir os nomes das músicas tocadas durante a viagem de Patrícia a partir da lista de comandos que ela executou no player.

É obrigatório o uso de **Listas Circulares Duplamente Encadeadas** para resolver o problema.

2) Exemplos de Entrada e Saída

A entrada consiste de um conjunto de casos de teste, cada um deles correspondente a uma playlist possível.

2.1) Entrada

A primeira linha de cada caso de teste contém um inteiro N indicando o número de operações/comandos realizados naquela playlist.

Cada uma das N linhas seguintes contém uma das operações/comandos descritas acima.

Após os N comandos executados, o programa lerá um novo valor N para executar novos comandos.

A execução do programa termina quando o número de operações de um caso de teste for 0 (**ZERO)**.

DICA: Para ler o nome das músicas, que pode ter espaços, uma das formas é a seguinte:

char titulo[128];

scanf(" %[^\t\n]s", titulo);

Note que há um espaço após as primeiras aspas dupla.

2.2) Saída

Para cada caso de teste, seu programa deve imprimir, na ordem de execução, os nomes de cada música executada.

Os nomes de música **devem ser separados por uma quebra de linha** (\n).

As saídas de cada caso de teste **devem ser separadas por duas quebras de linha (\n\n)**, com exceção do último caso de teste. O último caso terá **apenas uma única quebra de linha.**

Entrada	**Saída**
5 insere Como Nossos Pais insere Sujeito de Sorte inverte volta final 9 insere A Little Respect insere Always inverte insere Blue Savannah volta volta toca toca toca 0	Sujeito de Sorte Como Nossos Pais Blue Savannah Always A Little Respect

3) Dicas

- Para **compilar** seu código no terminal:
  - gcc lab.c -o lab
- **-o** significa output. Ele é responsável por gerar o binário do seu programa para execução. É OBRIGATÓRIO que o arquivo tenha a função **main;**
- Logo, o que você está dizendo é: *compile o código **lab.c** com o compilador **gcc**, gerando o executável (saída) **lab**;*

- Para **executar** seu programa:
  - ./lab

- Você pode baixar os arquivos de casos de teste do run.codes e executá-los manualmente:
  - ./lab < 01.in
- A diretiva < redireciona o conteúdo do arquivo 01.in para o terminal, cujas entradas/dados serão lidos pelo scanf;
- Você pode ainda redirecionar a saída impressa no terminal para um arquivo:

    - ./lab < 01.in > 01.res

- Por fim, você poder comparar sua reposta com o gabarito (resultado do caso de teste), fazendo
    - diff 01.res 01.out
    - onde 01.out é a saída esperada para a entrada 01.in

4) Observações Gerais

- A nota é dada pelo **número de casos de teste acertados**;
- É obrigatório usar **Listas Circulares Duplamente Encadeadas,** caso contrário, a nota será **ZERO**.
- É obrigatório desalocar a lista corretamente. Caso contrário, pontos serão descontados.
- Códigos com **erros de compilação e execução**, tais como Segmentation Fault, **serão considerados errados**;
- Utilize ***return 0;*** na main de seu programa;
- Qualquer tentativa de fraude, plágio e afins, corresponderá em **nota ZERO** para os envolvidos;