Lista de Exercícios Nº 1 (Manhã)

Esta lista é individual, seguindo as especificações contidas no arquivo `00_ProcListas.pdf` disponível no ambiente *Teams* da disciplina. A entrega ao professor deverá ocorrer até o dia 07 de abril de 2021.

1) Com base no programa criado em aula, que implementa uma lista linear de números por encadeamento, crie uma versão para manipular uma lista de funcionários. Cada funcionário possui um código (é um número do tipo `int`), um nome (*string* que pode ter até 30 caracteres úteis) e um salário (um número do tipo `float`). O programa deve permitir incluir, excluir e imprimir os funcionários. Nos *loopings* de inclusão e de exclusão, assuma que um código negativo indica o encerramento daquele processamento. Utilize um descritor da lista contendo a quantidade de funcionários e o valor total dos salários. Ao imprimir a lista, imprimir o somatório e a média dos salários. Separe a aplicação com um arquivo fonte para a parte cliente, um arquivo fonte para a implementação e um arquivo *header*.

2) Crie uma versão do programa do exercício 1 que implementa a lista por contiguidade. A exemplo do exercício 1, organize as funcionalidades em arquivo cliente, implementação e *header*. Considere que a lista precisará armazenar até 5000 funcionários.

3) Implemente um programa que constrói duas listas encadeadas de valores reais positivos e as imprime na tela. Encerrar a entrada de dados de cada lista assim que o primeiro valor negativo for informado. Após a entrada de dados, imprimir o conteúdo das duas listas. Em seguida, construir uma terceira lista encadeada correspondente à soma das duas listas originais, ou seja, uma lista onde o *n*-ésimo elemento contém a soma do *n*-ésimo elemento da primeira lista com o *n*-ésimo elemento da segunda lista, conforme o exemplo abaixo. Imprimir a nova lista na tela.

```
Lista 1:   2.00     3.25     4.75     1.20     0.12
Lista 2:  19.83     2.49     0.25     0.00
Lista 3:  21.83     5.74     5.00     1.20     0.12
```

4) Fazer um programa em C que recebe uma *string* de até 20 caracteres contendo uma expressão aritmética digitada pelo usuário. Essa expressão poderá conter 3 tipos de delimitadores de escopo e precedência: parênteses, colchetes e chaves. O programa deverá verificar se os delimitadores estão corretamente balanceados ou não, utilizando uma pilha para essa finalidade.

Exemplos de expressões válidas:

- `(A+B)/C`
- `A+(B/C)`
- `[a-2]*(5+c)`
- `{a*[c+d*(4*x)]}`
- `(A)`

Exemplos de expressões inválidas:

- `)A+B(/C`
- `A+B/C)`
- `[a-2)*(5+c]`
- `{a*[c+d*(4*x)}]`
- `(A`

Observações:

a) Considere que não há qualquer tipo de hierarquia entre os delimitadores, ou seja, o usuário pode, a qualquer momento, iniciar uma expressão com qualquer um dos 3 tipos permitidos.

b) Usar subrotinas para empilhar e desempilhar os elementos.

c) Use a pilha para nela acrescentar (apenas) os delimitadores de abertura, ou seja, '(', '[', '{' encontrados na *string*. Ao encontrar um delimitador de fechamento, ou seja, ')', ']', '}', retire o elemento do topo da pilha e verifique se ele é compatível com o delimitador que foi encontrado na cadeia. Se for, prossiga o processamento, se não for, encerre emitindo mensagem de erro. Se, ao final, a pilha estiver vazia, concluímos que a expressão contida na *string* é válida, pois está balanceada, e o programa deverá emitir uma mensagem informando isso.