Algoritmos e Estruturas de Dados 1

2023/2 - **Trabalho da disciplina**

Enunciado

Carlinhos está fascinado com o que está aprendendo em Programação! O garoto adora duas coisas em particular:

- Pilhas!, e
- Ordenar vetores com o *LSD RadixSort*, usando o *BucketSort* de 10 filas como método intermediário!

Carlinhos resolveu juntar as duas coisas, e pediu para você implementar uma *Pilha de vetores a serem ordenados*.

Escreva um programa em C que lê e processa vários comandos do usuário, onde cada comando pode ser:

- `push v0 v1 v2 ... -1`: ao ler o comando `push`, seu programa deve ler uma sequência de inteiros não negativos terminada em -1 (que não faz parte da sequência), formar um vetor com esses inteiros, e empilhar o vetor na pilha. Imprima `[v0 v1 v2 ...] empilhado.`.
- `pop`: ao ler o comando `pop`, seu programa deve desempilhar um vetor da pilha e ordená-lo com o *LSD RadixSort*, usando o *BucketSort* de 10 filas como método intermediário. Seu programa deve imprimir o vetor original e o vetor a cada passo do laço principal do método, conforme exemplo de saída abaixo. Se a pilha estiver vazia quando este comando for lido, imprima apenas `Pilha vazia.`.
- `F`: termina a execução do programa. Ao terminar a execução, imprima `N vetores sem ordenar.`, sendo N o número de vetores que sobraram na pilha.

Confira o seguinte exemplo de execução (o símbolo `>` indica entrada do usuário):

```
> pop
Pilha vazia.

> push 133 365 669 181 483 548 617 -1
[133 365 669 181 483 548 617] empilhado.

> push 15 83 35 44 -1
[15 83 35 44] empilhado.

> pop
[15 83 35 44] =>
[83 44 15 35] =>
[15 35 44 83].

> push 2734 8452 2098 1598 4236 7192 8689 5988 7147 5129 -1
[2734 8452 2098 1598 4236 7192 8689 5988 7147 5129] empilhado.

> push 591 354 -1
[591 354] empilhado.
```

```
> pop
[591 354] =>
[591 354] =>
[354 591] =>
[354 591].

> pop
[2734 8452 2098 1598 4236 7192 8689 5988 7147 5129] =>
[8452 7192 2734 4236 7147 2098 1598 5988 8689 5129] =>
[5129 2734 4236 7147 8452 5988 8689 7192 2098 1598] =>
[2098 5129 7147 7192 4236 8452 1598 8689 2734 5988] =>
[1598 2098 2734 4236 5129 5988 7147 7192 8452 8689].

> push 36 1 69 66 -1
[36 1 69 66] empilhado.

> pop
[36 1 69 66] =>
[1 36 66 69] =>
[1 36 66 69].

> push 282 6068 2193 6475 8173 9747 -1
[282 6068 2193 6475 8173 9747] empilhado.

> F
2 vetores sem ordenar.
```

Você pode assumir que todos os valores inseridos caberão em um *int* de 4 bytes; que os vetores terão no máximo 100 elementos cada; que a pilha terá no máximo 100 vetores durante toda a execução do programa; e que o usuário não entrará com outros comandos além dos especificados.

Implementação

O trabalho deve **obrigatoriamente** usar pilha(s) **e** fila(s) em sua solução. O trabalho deve conter os seguintes arquivos:

- `PE.h` e `PE.c`: definição e implementação de pilha usando como base um vetor ("pilha estática");
- `PD.h` e `PD.c`: definição e implementação de pilha usando como base uma lista ligada ("pilha dinâmica");
- `FE.h` e `FE.c`: definição e implementação de fila usando como base um vetor ("fila estática");
- `FD.h` e `FD.c`: definição e implementação de fila usando como base uma lista ligada ("fila dinâmica");
- `main.c`: programa principal. Deve incluir (via `#include`):
  - `PE.h` *ou* `PD.h`; *e*
  - `FE.h` *ou* `FD.h`.

O programa principal deve utilizar filas e pilhas como estruturas *abstratas* de dados. Em particular, deve ser possível "escolher" entre usar pilhas estáticas ou dinâmicas *apenas alterando os* `#include` *e recompilando de acordo*! Da mesma forma, deve ser possível "escolher" entre usar filas estáticas ou dinâmicas de maneira análoga (note que, desta forma, há um total de quatro "configurações" com as quais o trabalho deverá funcionar).

Independentemente da implementação, certifique-se que toda memória alocada por seu programa é desalocada ao final da sua execução.

Orientações

- O trabalho pode ser feito por equipes de *no máximo* 2 (dois) estudantes;
- Submeta, via *Moodle*, um pacote **zip** ou **tar.gz** contendo os 9 arquivos citados acima, além de um arquivo de texto (txt) onde conste:
  - O nome de todos os integrantes da equipe;
  - Toda informação que a equipe julgar relevante para a correção (como *bugs* conhecidos, detalhes de implementação, escolhas de projeto, etc.)
- Comente adequadamente seus códigos para facilitar a correção;
- Atenção: a correção será parcialmente automatizada, e a saída do programa será testada com outras entradas além das fornecidas como exemplo. *Siga **fielmente** o formato de entrada e de saída dado nos exemplos*, sob pena de grande redução da nota;
- Certifique-se que seu programa compila e funciona antes de submetê-lo;
- O trabalho deve ser entregue até **6 de Dezembro de 2023, 23:59**, apenas via *Moodle*. Trabalhos entregues por outros meios ou fora do prazo não serão aceitos. É suficiente que o trabalho seja submetido por apenas um estudante da equipe;
- Trabalhos detectados como cópia, plágio (de colegas ou da internet) ou comprados receberão **todos** a nota 0 (**ZERO**) e estarão sujeitos a abertura de Processo Administrativo Disciplinar Discente.