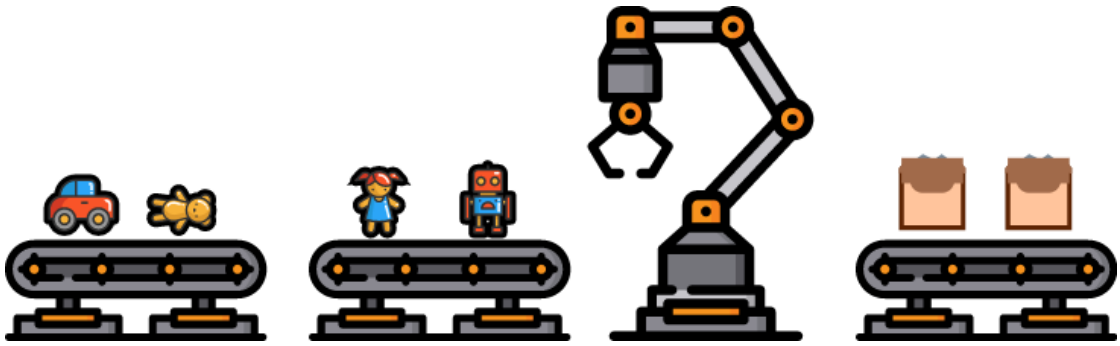
UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação - ICMC
SCC 202/502 – Algoritmos e Estrutura de Dados I - 2º Sem /2018
Prof. Robson Cordeiro
PAE: Guilherme Queiroz, Leonardo Moraes

Trabalho 1: Loja de Brinquedos

Entrega até 24/10/2018, via atividade do Tidia.

Você foi contratado para programar um robô de uma fábrica que é responsável por armazenar n brinquedos que estão em uma esteira de produção, organizados como uma lista circular. Esse robô deverá percorrer a esteira e, a cada k brinquedos percorridos, pegar o k-ésimo brinquedo e empilhá-lo dentro de uma caixa. Observa-se que o valor de k deve ser decrescido de 1 sempre que o final da lista for alcançado, e cada caixa de brinquedos pode conter no máximo p brinquedos; quando uma caixa enche, deve-se inseri-la em uma fila de caixas e criar uma nova caixa para alocar os demais brinquedos. Inicialmente, cada caixa criada está vazia.
- Repita todo o procedimento de selecionar um brinquedo e empilhá-lo até que não haja mais brinquedos na esteira. Lembre-se de selecionar os brinquedos de k em k, e de decrescer k sempre que necessário.
- As variáveis, n quantidade de brinquedos na esteira, k brinquedos a serem percorridos e p quantidade máxima de brinquedos por caixa devem ser informadas pelo usuário.



Após organizar todas as caixas, seu programa deverá remover as caixas da fila, uma a uma, imprimindo o brinquedo que está no topo da caixa (pilha) e a quantidade de brinquedos presente.
Exemplo: caixa *x*: brinquedo *y* - quantidade *z*
- *x*: número da caixa, começa pelo número 1;
- *y*: dados do brinquedo no topo;
- *z*: quantidade de brinquedos dentro da respectiva caixa.

1. Objetivos

Seu programa deverá receber como parâmetro da função *main* a quantidade total de brinquedos (*n*), o intervalo (*k*) e a capacidade (*p*) de cada caixa, além de uma sequência com *n* pares <número de série, modelo> descrevendo cada brinquedo. Crie uma lista circular para representar a esteira. O TAD correspondente deve conter **pelo menos** as seguintes operações:

- cria_lista: cria uma lista vazia;
- insere_na_lista(brinquedo): insere um brinquedo (isto é, um struct com o número de série e o modelo do brinquedo) no final da lista;
- remove_da_lista(i): remove e retorna o brinquedo na posição i da lista;
- vazia_lista(): retorna "true" se a lista estiver vazia; "false", caso contrário;
- cheia_lista(): retorna "true" se a lista estiver cheia; "false", caso contrário. Deve existir apenas para a implementação estática;

Crie uma pilha para representar cada caixa de brinquedos. O TAD correspondente deve conter **pelo menos** as seguintes operações:

- cria_pilha: cria uma pilha vazia;
- push(brinquedo): insere um brinquedo na pilha/caixa, segundo a ordenação LIFO, em que o último a entrar é o primeiro a sair;
- top(): retorna o brinquedo no topo da pilha/caixa, sem removê-lo;
- cheia_pilha(): retorna "true" se o número de brinquedos for igual a *p*; "false", caso contrário;

Crie uma fila de caixas de brinquedos, isto é, uma fila de pilhas. O TAD correspondente deve conter **pelo menos** as seguintes operações:

- cria_fila: cria uma fila vazia;
- insere_na_fila(caixa): insere uma caixa (isto é, uma pilha de brinquedos) na fila, segundo a ordenação FIFO, em que o primeiro a entrar é o primeiro a sair;
- remove_fila(): remove uma caixa da fila;
- vazia_fila(): retorna "true" se a fila estiver vazia; "false", caso contrário;
- cheia_fila(): retorna "true" se a fila estiver cheia; "false", caso contrário. Deve existir apenas para a implementação estática;

Deve-se implementar **duas versões** do sistema:

- Estática sequencial (**peso 3,5**);
- Dinâmica encadeada (**peso 6,5**).

Observações

- Insira todos os brinquedos na esteira, e percorra-a partindo do primeiro brinquedo;
- n: maior ou igual a 1 e menor ou igual a 1000 - *escala* $[1, 1000]$- *tipo inteiro;*
- k: maior ou igual a 1 e menor ou igual a n - *escala* $[1, n]$ - *tipo inteiro;*
- p: maior ou igual a 1 e menor ou igual a 1000 - *escala* $[1, 1000]$- *tipo inteiro.*

Todas as estruturas de dados pedidas devem ser implementadas por completo; **não é permitido** o uso de bibliotecas (ou similares) que incluam implementações totais ou parciais das mesmas.

Formatação do Programa

Entrada: n k p num_s_b1 mod_b1 num_s_b2 mod_b2 ... num_s_bn mod_bn
Saída:
caixa x_1: brinquedo y_1 - quantidade z_1
caixa x_2: brinquedo y_2 - quantidade z_2
....
caixa x_m: brinquedo y_m - quantidade z_m

Exemplo de Execução

Comando: ./exec 5 3 2 1 carro 2 boneca 3 urso 4 carro 5 boneca

Execução:

- k = 3;
- Adiciona os 5 brinquedos na esteira.
- Lista: <1,carro> <2,boneca> <3,urso> <4,carro> <5,boneca> => retira <1,carro>, adicionar na caixa 1;
- Lista: <2,boneca> <3,urso> <4,carro> <5,boneca> => retira <4,carro>, adicionar na caixa 1;
- Fim da lista, k = 2;
- Lista: <2,boneca> <3,urso> <5,boneca> => retira <2,boneca>, caixa 1 cheia, adicionar na caixa 2;
- Lista: <3,urso> <5,boneca> => retira <5,boneca>, adicionar na caixa 2;
- Fim da lista, k = 1;
- Lista: <3,urso> => retira <3,urso>, caixa 2 cheia, adicionar na caixa 3;
- Terminou.

Saída:
caixa 1: brinquedo <4,carro> - quantidade 2
caixa 2: brinquedo <5,boneca> - quantidade 2
caixa 3: brinquedo <3,urso> - quantidade 1

---

2. Indicações Gerais

• O trabalho deve ser desenvolvido em **duplas** de alunos.
• Serão aceitos somente trabalhos em linguagem de programação C.

---

3. Especificações

Implemente as estruturas pedidas (lista circular, pilha e fila de pilhas) utilizando os conceitos de TAD e seguindo os critérios detalhados na Seção 4; caso necessário, implemente

estruturas adicionais de sua escolha. Para questões específicas que não estejam detalhadas, usem o bom senso e documentem suas decisões no relatório a ser entregue. Caso necessitem, entrem em contato com os estagiários PAE ou com o professor da disciplina para tirar eventuais dúvidas.

---

4. Critérios de Avaliação

Os trabalhos serão avaliados em acordo com os seguintes critérios:
• 4.1. Padronização: o programa deve respeitar estritamente os padrões de entrada e saída;
• 4.2. Corretude: o programa deve fazer o que foi especificado;
• 4.3. Estruturas de dados utilizadas: adequação e eficiência;
• 4.4. Observação de “boas práticas” de programação: TAD, modularidade do código, documentação interna, indentação, etc.

Observação
• Plágio de programas ou de material adicional não será tolerado; será utilizada **ferramenta automática** de detecção de plágio, além de análise manual.
*Qualquer material similar ao de outra dupla terá **nota zero** independentemente de qual for a cópia.*

---

5. Entrega do Trabalho

A entrega dos trabalhos será via **Atividade do Tidia**. São requeridos:
• 5.1. Um arquivo comprimido (ZIP) que deverá ser postado por apenas um dos membros do grupo, no site da disciplina, e deve conter (a) arquivo ‘readme.txt‘ com nomes e números USP dos integrantes do grupo; (b) arquivos de código-fonte do programa; e (c) arquivo ‘howtodo.txt‘ com passo a passo de como compilar o programa (deixar claro que compilador usou, versão, se utilizou alguma biblioteca diferente, qual sistema operacional utilizou, etc).
• 5.2. Relatório breve (documentação externa) do programa de, no máximo, 2 páginas, contendo, pelo menos, (a) breve descrição do trabalho feito; (b) apresentação das estruturas de dados utilizadas e justificativas para a utilização de todas as estruturas adicionais utilizadas; e (c) apresentação das diferenças na utilização de abordagens de estruturas estáticas e dinâmicas no trabalho (qual é a melhor considerando as características da aplicação?).

Entregas em atraso: a nota máxima do trabalho será decrescida de 1.0 ponto ponto por dia de atraso na entrega.