MAC-2301 — Estruturas de Dados

Primeira lista de exercícios

1. Considere um estacionamento de trens (analogia com a pilha), como visto em aula.

   (a) Suponha que existam seis trens na entrada, numerados por $1, 2, 3, 4, 5$ e 6. Os trens poderão sair na ordem 325641? E na ordem 154623? (Justifique).

   (b) Escreva um algoritmo que dado $n > 0$ e dada uma permutação de $1, 2, \ldots, n$, decide se os trens poderão sair na ordem especificada na permutação dada.

2. Considere o problema do exercício anterior substituindo a pilha por uma fila dupla.

   (a) Determine uma permutação de $1, 2, 3$ e 4 que possa ser obtida com uma fila dupla com entrada restrita, mas que não possa ser obtida com uma fila dupla com saída restrita.

   (b) Determine uma permutação de $1, 2, 3$ e 4 que possa ser obtida com uma fila dupla com saída restrita, mas que não possa ser obtida com uma fila dupla com entrada restrita.

   (c) Determine uma permutação de $1, 2, 3$ e 4 que não possa ser obtida com uma fila dupla com entrada restrita e nem com uma fila dupla com saída restrita.

   Justifique cada um dos casos.

3. Reescreva a função abaixo utilizando uma **pilha** para eliminar, isto é, simular a recursão:

   (a)
   ```
   int Comb(int n,int m){
      if ((n==1) || (m==0) || (m==m))
        return 1;
      else
        return Comb(n-1,m) + Comb(n-1,m-1);
   }
   ```

4. Considere o problema de uma pilha e uma fila linear em alocação seqüencial armazenadas num vetor `X` de tamanho M.

   (a) Explique como seria a sua implementação:
   - Represente graficamente uma situação genérica.
   - Descreva como inicializar cada um dos apontadores.

   (b) Escreva os algoritmos para inserção e remoção dos elementos da pilha.

   (c) Escreva os algoritmos para inserção e remoção dos elementos da fila.

   (d) Escreva os algoritmos para tratamento de *overflow* na pilha e na fila.

   Obs: Suponha existentes as funções *UNDERFLOW* e *OVERFLOW*. Lembre-se que a função *OVERFLOW* só deve ser chamada quando não houver mais espaço disponível no vetor.

5. Faça as seguintes funções para a seguinte estrutura de lista ligada:

```
typedef struct no1{
           int info;
           struct no1 *prox;
        } node, *apontnode;
```

   (a) Uma função que dada uma sequência terminada por zero devolve uma lista ligada com os elementos lidos;
   (b) Faça uma função que insere um elemento no início da lista;
   (c) Faça uma função que dados p remove o elemento na posição p (a partir do primeiro elemento) da lista;
   (d) Faça uma função onde dadas duas listas de números ordenadas retorna uma única lista também ordenada;
   (e) Faça uma função que dada uma lista remove o seu nó com o maior valor;
   (f) Faça uma função que ordena uma lista (dica: observe quais itens anteriores você pode usar...).