IMD0029 - Estrutura de Dados Básicas 1 – Turma 04 – 2016.2 – Prova de Reposição
Prof. Eiji Adachi M. Barbosa

Nome:_____

Matrícula: _____

ANTES DE COMEÇAR A PROVA, leia atentamente as seguintes instruções:

- Esta é uma prova escrita de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A prova vale 10,0 pontos e o valor de cada questão é informado no seu enunciado.
- Preze por respostas legíveis, bem organizadas e simples. Não é necessário seguir a notação de C++, embora seja permitido usar tal notação. Pode usar uma notação mais simples, estilo pseudo-código.
- Leia atentamente às observações em cada questão. Elas fazem parte do enunciado.
- As respostas devem ser fornecidas preferencialmente em caneta. Respostas fornecidas a lápis serão aceitas, mas eventuais questionamentos sobre a correção não serão aceitos.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- Desvios éticos ou de honestidade levarão à anulação da prova do candidato (nota igual a zero).

Questão 1: (2,0 pts) Um quadro de Young é uma matriz de ordem m x n contendo números naturais sem repetição a qual segue as seguintes propriedades: (i) os elementos de cada linha estão em sequência ordenada crescente da esquerda para a direita e (ii) os elementos das colunas estão em sequência ordenada crescente de cima para baixo. A matriz abaixo representa um quadro de Young:

9	15	16	20
10	21	23	25
17	30	33	39
18	31	34	51

Neste contexto, implemente uma função que verifica se um dado número X está armazenado ou não num quadro de Young. Em outras palavras, sua função deverá retorna *true* se X está armazenado num determinado quadro de Young e *false* caso contrário. Sua função deverá obrigatoriamente ter complexidade de ordem assintótica em termos de tempo de execução de $O(m*lg(n))$. Considere que o quadro de Young é representado como uma matriz de inteiros de ordem m x n. Sua função deverá seguir a seguinte assinatura:

```
bool Young( int quadro[m][n], int x)
```

Questão 2: (3,0 pts) Implemente uma função de Partição que opere sobre uma lista duplamente encadeada com sentinelas cabeça (*head*) e cauda (*tail*). Considere que a lista de entrada não contém elementos repetidos. Sua função deverá seguir a seguinte assinatura:

```
void Partition( List l );
```

Para esta questão, considere que a função `Partition` irá operar sobre as seguintes estruturas que definem uma *Lista Duplamente Encadeada com Sentinelas*:

```
List {
     Node head;
     Node tail;
}
```

```
Node {
     int value;
     Node next;
     Node previous;
}
```

Obs.1: É proibido modificar as estruturas do enunciado. Também é proibido criar uma outra lista ou um vetor auxiliar na solução desta questão. Ou seja, é obrigatório operar apenas sobre a lista passada como parâmetro. Questões que desobedeçam tais instruções serão desconsideradas.

Questão 3: (3,0 pts) Um algoritmo de ordenação é dito estável se não altera a posição relativa dos elementos que têm um mesmo valor. Em outras palavras, se dois elementos possuem valores iguais e a ordem relativa em que eles aparecem na sequência de dados inicial é mantida após a ordenação, então este algoritmo de ordenação é estável. Considere como exemplo o seguinte vetor v[1..5] = {1', 2', 3´, 1, 2, 3}. Se após a execução de um algoritmo de ordenação o vetor encontrar-se da seguinte forma: v = {1', 1, 2', 2, 3', 3}, então este algoritmo de ordenação é dito estável. Perceba que os números repetidos não alteraram as posições relativas que tinham antes da ordenação (elementos repetidos com aspas simples apareciam primeiro do que os repetidos sem aspas antes da ordenação e continuaram aparecendo primeiro após a ordenação). Neste contexto, considere o seguinte algoritmo de intercalação:

```
1.   Intercalar( v[n], inicio1, inicio2, fim2 ) :
2.       DECLARE tmp[n]
3.       fim1 = inicio2-1, i = inicio1, j=inicio2, k=0
4.       ENQUANTO i ≤ fim1 E j ≤ fim2 FAÇA:
5.             SE v[i] < v[j] ENTÃO:
6.                   tmp[k] = v[i],i = i+1
7.             SENÃO:
8.                   tmp[k] = v[j], j = j+1
9.             FIM_SE
10.            k = k+1
11.      FIM_ENQUANTO
12.      ENQUANTO i ≤ fim1 FAÇA:
13.            tmp[k] = v[i], i = i+1, k=k+1
14.      FIM_ENQUANTO
15.      ENQUANTO j ≤ fim2 FAÇA:
16.            tmp[k] = v[j], j = j+1, k=k+1
17.      FIM_ENQUANTO
18.      COPIE tmp EM v
19.  FIM
```

Suponha que você vai usar o algoritmo acima como parte do seu algoritmo de ordenação por intercalação (*mergesort*). O seu algoritmo de ordenação será estável? Se sim, o que garante sua estabilidade? Se não, o que precisa ser mudado para garantir sua estabilidade?

Questão 4: (3,0 pts) Um anagrama é um jogo de palavras em que tentamos criar outras palavras por meio do rearranjo das letras de uma determinada palavra base. As novas palavras criadas devem empregar todas as letras originais da palavra base exatamente uma vez. Um exemplo bastante conhecido de anagrama é o nome da personagem Iracema, do escritor José de Alencar, que é um anagrama da palavra America. Nesta questão, você implementará uma solução que irá auxiliar uma pessoa a encontrar anagramas de uma determinada palavra. Tal solução terá como base um dicionário de anagramas. Para construir este dicionário, implemente uma função que irá receber como entrada um *array* de palavras que servirão como base para a sua solução "aprender" possíveis anagramas e construir o dicionário desejado. Esta função deverá ter a seguinte assinatura:

```
Dictionary learn(string words[], int arraySize)
```

Faz parte da solução desta questão a definição da estrutura `Dictionary`. Portanto, além de implementar a função `learn`, você também deverá mostrar sua definição da estrutura `Dictionary`. Não é necessário implementar as operações básicas sobre o dicionário (Inserir, Remover e Buscar); pode considerar que estas operações já estão implementadas e podem ser reusadas. O que é requerido nesta questão é a definição da estrutura interna do dicionário. Ou seja, quais seus atributos e tipos de cada atributo.

Obs.1: Nesta questão, você poderá empregar funções auxiliares de alto nível, desde que fique claro qual o propósito destas funções, seja por meio de comentários, ou pelo próprio nome da função.