IMD0029 - Estrutura de Dados Básicas 1 – 2017.1 – Prova 01
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 obrigatório seguir a sintaxe de C++; pode usar uma sintaxe mais simples estilo pseudo-código.
- 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: (3,0 pontos) Harry Potter está em apuros mais uma vez em Hogwarts. Harry foi pego após roubar ingredientes do armário da sala do professor Snape e, por isso, está na sala de detenção de alunos. Como parte de sua punição, Harry foi obrigado a ordenar todas as milhares receitas de poções do professor Snape. O professor Snape exigiu que as receitas ficassem ordenadas da seguinte forma: primeiro, devem estar em ordem crescente do ano de publicação; depois, para receitas do mesmo ano, devem estar em ordem decrescente do nome. Num momento em que o professor Snape ausentou-se da sala, Harry Potter levantou sua varinha e lançou o feitiço *"wingardium sortita"*. Porém, nada aconteceu com as velhas receitas do professor Snape. Harry lamentou-se amargamente de não ter prestado atenção nas aulas de algoritmos mágicos. Ele olha para o lado e pede sua ajuda. Infelizmente, você também não lembra o feitiço para ordenar magicamente vários pergaminhos. Por sorte, você está matriculado na disciplina de programação e estruturas de dados mágicas e lembra-se dos algoritmos de ordenação e, por isso, decide ajudar Harry. Neste contexto, projete um algoritmo de ordenação baseado no ***Quick Sort*** para ajudar Harry Potter a ordenar as receitas do professor Snape. Para esta questão, sua solução deverá ter a assinatura definida abaixo. Considere também que as receitas são representadas pela estrutura abaixo.

```
void Ordenar( Receita v[], int tamanho );
```

```
Receita {
  int Ano;
  string Nome;
  List igredientes;
  List passo-a-passo;
}
```

Obs.1: Considere que o atributo do tipo string pode ser comparado da mesma forma que atributos do tipo int, isto é, usando os operadores "<", "<=", ">", ">=", "==" e "!=".

Questão 2: (3,0 pontos) Severo Snape é um professor de Hogwarts com características pessoais bastante peculiares. Ele possui transtorno obsessivo compulsivo e gosta de manter todas suas coisas em ordem. Por este motivo, costuma ser bastante impaciente com alunos bagunceiros em sala de aula. Na sua sala, todos os ingredientes para poções mágicas ficam dispostos num grande array mágico flutuante em ordem crescente do nome do ingrediente. Numa das aulas de poções mágicas do professor Snape, os irmãos gêmeos Fred e Jorge Weasley decidem pregar uma peça. Eles pegam os últimos M potes do array de ingredientes e os colocam em ordem decrescente. Desta forma, o array de ingredientes fica com uma sequência inicial crescente e uma sequência final decrescente. Na aula seguinte, após perceber a bagunça no seu array mágico de ingredientes, o professor Snape fica furioso, mas oferece mil pontos para a casa do aluno que for capaz de desfazer a bagunça. Como bons programadores bruxos, você, Harry, Hermione e Rony decidem criar um algoritmo para resolver o problema. Depois de divididas as tarefas, você ficou responsável por implementar o algoritmo que identifica quantos M potes foram desordenados pelos gêmeos Weasley. Nesta questão, faça uma solução iterativa com complexidade $\Theta(\log_2 n)$ para definir quantos M potes foram desordenados. Sua solução deverá seguir a assinatura: `void identificar_M( Ingrediente v[], int tamanho );`

Questão 3: (2,0 pontos) Num de seus estudos para a prova de programação e estruturas de dados mágicas, Harry Potter se deparou com o algoritmo de ordenação por inserção (*Insertion Sort*). Nos pergaminhos sobre algoritmos de ordenação, Harry leu que o algoritmo *Insertion Sort* funciona da seguinte forma: *"O algoritmo de ordenação por inserção funciona mantendo num array v[N] duas regiões distintas: uma região que fica entre [0...i], onde ficam os elementos já ordenados, e uma outra região entre [i+1...N-1], onde ficam os elementos não ordenados. A cada iteração, o algoritmo pega o primeiro elemento na região não ordenada, busca a posição onde inseri-lo e coloca-o corretamente na região ordenada".* Harry então se lembra das aulas de algoritmos de busca e pensa: *"Posso melhorar a eficiência do algoritmo Insertion Sort se eu usar a busca binária para buscar a posição onde inserir o novo elemento na região ordenada".* Você concorda com a afirmação de Harry Potter? Justifique. Apresente também um pseudo-código em altíssimo nível de abstração do *Insertion Sort* para justificar sua resposta.

Questão 4: (2,0 ponto) Para cada uma das afirmações a seguir, marque V (verdadeiro) ou F (falso), justificando sua resposta.

( ) Os algoritmos *Quick Sort*, e *Merge Sort* possuem a mesma complexidade assintótica para o melhor e pior caso.

( ) Os algoritmos *Insertion Sort*, *Bubble Sort* e *Selection Sort* possuem a mesma complexidade assintótica para o melhor e o pior caso.

( ) Os algoritmos de busca binária e busca sequencial possuem a mesma complexidade assintótica para o melhor e pior caso.

( ) O *Merge Sort* é um algoritmo dividir-pra-conquistar em que cada etapa de divisão o vetor de entrada é sempre dividido em duas partes de mesmo tamanho, ou com tamanhos de diferença igual a 1.

( ) O *Quick Sort* é um algoritmo dividir-pra-conquistar em que cada etapa de divisão o vetor de entrada é sempre dividido em duas partes de mesmo tamanho, ou com tamanhos de diferença igual a 1.