Universidade de Brasília
Departamento de Ciência da Computação
Professor: Genaína Nunes Rodrigues
Créditos: prof. Alessandro Garcia (PUC-Rio)
Nome dos alunos / Matrícula de cada aluno:
1.
2.

Observações:

- Nas questões dessa lista, os algoritmos poderão ser apresentados em C ou em pseudocódigo. Desconsidere qualquer eventual erro sintático (ex. ausência de ponto-e-vírgula, etc).

- Em relação as questões, não deve ser feito nenhuma suposição sobre qualquer dado (de entrada, de saída, estruturas de dados, campos de estruturas, etc.) apresentado exceto quando for mencionado. Portanto, não assuma qualquer restrição desde que esta esteja explicitamente declarada na questão.

- Ao descrever as assertivas, caso a dupla use uma notação diferente dos símbolos lógicos e dos operadores tradicionais de linguagens de programação, a dupla deverá explicar a notação. As assertivas também poderão ser descritas textualmente desde que sua descrição seja clara e compreensível.

- Ao descrever erros originados por quebras de assertivas, assuma que existe uma função “ExibirErro(String msg)” a qual recebe e exibe uma mensagem (String). Essa mensagem deverá indicar quais condições foram quebradas.

Lista de Exercícios sobre Assertivas

1. O que são assertivas de entrada? Assertivas de saída? Dentro de um projeto de sistema baseado em **Contratos**, qual seriam as responsabilidades do **cliente** e do **servidor**? (1 ponto)

2. Considere a existência de uma função para registrar salários de funcionários de uma dada empresa num sistema. A interface da função e as assertivas de entrada são descritas na área retangular a seguir abaixo. Considere a existência também de uma função booleana `ExisteFuncionario (int id)` o qual retorna verdadeiro se existir um funcionário registrado no sistema com identificador igual a id e falso caso contrário.

```
/* Assertivas de Entrada:
    *   idPessoa deve ser o identificador de uma pessoa
registrada no sistema
    *   salario deve ser diferente de null
    *   salario deve ser menor que 3000 e maior 500
    *   se o salario for menor que 1000, o cargo devera ser
'secretario'
    *   se o salario for maior que 1000 e menor que 2000, o
cargo devera ser 'desenvolvedor'
    *   se o salario for maior que 2000, o cargo devera ser
'chefe'
    *   para todos os outros casos de valores da variavel
salario, o cargo devera ser 'desconhecido'
    */
   void RegistrarSalario (int idPessoa, double salario,
char * cargo);
```

Considere o código cliente abaixo que faz chamada a função para registro de salário. Altere o código cliente de forma que ele cumpra as regras de **Contrato** estabelecidas na função `RegistrarSalário`. Utilize assertivas executáveis. (1 ponto)

```
public void RegistrarFuncionario (double salario, char*
nome, char * cargo) {
        int idFuncionario = ObterId();
        RegistrarNome (idFuncionario, nome);
        RegistrarSalario(idFuncionario, salario, cargo);
   }
```

3. Considere o algoritmo abaixo retirado do livro do Cormen. No procedimento MERGE, O parâmetro *A* é um vetor de inteiros e *p*, *q* e *r* são índices de enumeração dos elementos desse vetor tal que $p <= q < r$. Assuma que os subarranjos *A[p..q]* e *A[q+1..r]* estão em seqüência ordenada crescente. Assumindo que a entrada *A* é definida e *p*, *q* e *r* respeitam a equação descrita anteriormente, elabore assertivas de saída com relação ao vetor. Explore as diversas características de um vetor considerando o seu tamanho, os elementos que compõe o vetor, os valores e as posições desses elementos. Lembre-se que as assertivas de saídas podem referenciar estados e valores das variáveis de entrada. Faça assertivas para os valores *L[i]* e *R[j]* ao final da execução algoritmo. (2 pontos)

```
MERGE(A, P, q, r)
 1  n1 ← q − p + 1
 2  n2 ← r − q
 3  criar arranjos L[1..n1 + 1] e R[1..n2 + 1]
 4  for i ← 1 to n1
 5       do L[i] ← A[p + i − 1]
 6  for j ← 1 to n2
 7       do R[j] ← A[q + j]
 8  L[n1 + 1] ← ∞
 9  R[n2 + 1] ← ∞
10   i ← 1
11   j ← 1
12   for k ← p to r
13        do if L[i] ≤ R[j]
14             then A[k] ← L[i]
15                  i ← i + 1
16             else A[k] ← R[j]
17                  j ← j + 1
```

4. Fulano deseja fazer um sistema que o auxilie a marcar bilhetes de aposta da Mega Sena. Ele utiliza uma matriz de inteiros chamada *tabela* para representar o bilhete de aposta. Um dado inteiro *tabela[x][y]* apresentará valor 0 caso o número com dezena *x* e unidade *y* não seja marcado para aposta e 1 caso contrário. A tabela apenas poderá conter valores 0 e 1 para os números que podem ser apostados. Os números presentes na tabela de aposta variam dos inteiros presentes no intervalo de 0 até 59. Em uma aposta, no máximo 6 números poderão ser escolhidos.

   Considere a função abaixo responsável por realizar a marcação de aposta em um número no bilhete. Ela recebe a dezena e a unidade do número a ser apostado e a tabela de aposta. Elabore assertivas executáveis que verifiquem as pré-condições a serem satisfeitas para que o procedimento atenda as restrições descritas acima e realiza a aposta no dado número. Note que a função só irá de fato fazer a marcação de forma apropriada caso certas pré-condições forem satisfeitas. (3 pontos).

```
void ApostarNumero (int dezena, int unidade, int tabela[6]
[10]) {
        tabela[dezena][unidade] += 1;
   }
```

5. Considere um sistema que realiza a média salarial de uma escola. A escola contém três categorias de emprego: professor, diretor e faxineiro. Os dados de cada funcionário são armazenados numa lista duplamente encadeada. Assim a lista apresenta um ponteiro para o elemento corrente (*pElemCorr*), ponteiros para a origem e fim da lista, número de elementos e todas as funções presentes no módulo LISTA. Cada nó da lista apresenta o salário de cada funcionário através do campo *salário* e sua categoria através do campo *categoria*. O campo categoria é um inteiro. Para professores, o seu valor deve ser 0, para diretores 1 e para faxineiros 2.

Considere a existência da função *Exibir_Medias* que é responsável por exibir as medias de cada categoria.

O código na última página dessa lista apresenta o procedimento responsável por calcular as médias salariais e exibir o resultado das mesmas. Note que o procedimento itera sobre a lista principal. Para que esse procedimento funcione corretamente, várias assertivas devem ser satisfeitas. Caso essas assertivas não forem satisfeitas o procedimento não ter sucesso no calculo da média, apresentando faltas.

Assuma que a lista passada segue as propriedades de uma lista duplamente encadeada. Considere que as constantes `NUMERO_PROFESSORES`, `NUMERO_DIRETORES` e `NUMERO_FAXINEIROS` foram declaradas.

Liste as assertivas sobre o parâmetro *pListaPrincipal* para que o procedimento funcione corretamente. Liste a propriedade invariante a respeito das constantes que representam a quantidade de funcionários de cada categoria para que o procedimento seja executado sem problemas. Identifique as faltas que podem ocorrer devido a ausência de verificações sobre as assertivas que você apresentou e indique como elas poderiam ser verificadas. (3 pontos)

```
typedef struct tagFuncionariosLista {

        int salario;
               /* salario do funcionario*/

        int categoria;
               /* categoria do funcionario: professor(0), diretor(1),
faxineiro(2) */

         struct tagFuncionariosLista * pAnt ;
               /* Ponteiro para o elemento predecessor */

         struct tagFuncionariosLista * pProx ;
               /* Ponteiro para o elemento sucessor */

   } tpFuncionariosLista ;

void ExibirMediaSalarial(LIS_tppLista pListaPrincipal) {

       int mediaSProf = 0; // guarda media salario dos professores
       int mediaSDir = 0; // guarda media salario dos diretores
       int mediaSFaxi = 0; // guarda media salario dos faxineiros

       int i;
       for( i = 0; i < NUMERO_PROFESSORES; i++ )
          {
                 mediaSProf += pListaPrincipal->pElemCorr->salario;
            // avança a posicao do Elem Corrente em 1
            LIS_AvancarElementoCorrente( pListaPrincipal, 1 ) ;
          } /* for */

       mediaSProf = mediaSProf/ NUMERO_PROFESSORES;

       for( i = 0; i < NUMERO_DIRETORES; i++ )
          {
                 mediaSDir += pListaPrincipal->pElemCorr->salario;
            // avança a posicao do Elem Corrente em 1
            LIS_AvancarElementoCorrente( pListaPrincipal, 1 ) ;
          } /* for */

       mediaSDir = mediaSDir/ NUMERO_DIRETORES;

       for( i = 0; i < NUMERO_FAXINEIROS; i++ )
          {
                 mediaSFaxi += pListaPrincipal->pElemCorr->salario;
            // avança a posicao do Elem Corrente em 1
            LIS_AvancarElementoCorrente( pListaPrincipal, 1 ) ;
          } /* for */

       mediaSFaxi = mediaSFaxi/ NUMERO_FAXINEIROS;

       //Exibi os valores obtidos das medias salarias
       Exibir_Medias(mediaSProf, mediaSDir, mediaSFaxi);
    }
```