Instituto de Informática – INF
Cursos de Bacharelado do INF/UFG (NBC)

Avaliação

Disciplina:	**Algoritmos & Estruturas de Dados – 1**	
Professor:	Wanderley de Souza Alencar	**Nota:**
Aluno(a):		**Matrícula:**
2ª Avaliação Formal – 2020/1		**Data: 17/12/2020**

INSTRUÇÕES PARA RESOLUÇÃO DESTA AVALIAÇÃO

1. Esta avaliação possui 04 (quatro) questões, todas com valor igual a 5,0 (cinco) pontos;
2. É permitido resolver qualquer quantidade de questões para atingir pontuação máxima: 10,0 (dez) pontos;
3. As resoluções devem ser submetidas à área da disciplina no *Sharif Judge System*, cujo endereço é: <https://sharif.inf.ufg.br/wanderley>. Neste sistema, cada questão corresponde a 500 (quinhentos) pontos e, portanto, atingindo-se 1000 (mil) pontos você obteve pontuação máxima nesta avaliação: 10,0 (dez);
4. As submissões das resoluções devem ser realizadas até às **23h59min do dia 18/12/2020 (sexta-feira)**. Não é permitido o envio de resoluções por *e-mail*;
5. As notas desta avaliação serão publicadas até o dia 22/12/2020 e, após publicação, será franquado prazo de 24h (vinte e quatro horas) para recorrer da nota atribuída. Isto deve ser feito enviando-se *e-mail* para o professor (`wanderleyalencar@ufg.br`);
6. Até às 18h do dia 23/12/2020 a nota final desta avaliação (ou seja, após avaliação dos recursos) será publicada na área da disciplina na Plataforma Turing e será **definitiva**.

Questão 01 (5,0 pontos) Você está, neste momento, estagiando no *Centro de Tecnologia da Informação* de uma rede de supermercados e, numa conversa com um(a) profissional que atua na área de controle de estoque, ouviu o seguinte relato dele:

"Uma das atividades mais *chatas* que tenho que fazer frequentemente é a de comparar duas grandes listas para dizer se elas são, ou não, iguais. Cada uma destas listas contém os códigos dos produtos armazenados em dois almoxarifados desta rede de supermercados.

O *chato* é que os códigos dos produtos, nas duas listas, estão todos fora de ordem e podem ainda, cada um, aparecer mais de uma vez.

Duas listas somente são consideradas *iguais* se contiverem os mesmos códigos de produtos, independentemente do número de ocorrências deles em cada uma das listas.

Diante disso, às vezes consumo todo um dia de trabalho para fazer esta conferência manualmente."

Você então disse a ele(a): não se preocupe, vou elaborar um programa $\mathbb{C}$ para resolver este seu problema em segundos, desde que você informe as duas listas que deseja comparar. E vou além: se elas forem diferentes, apresentarei os itens que as diferem.

O(A) profissional disse-lhe: você se tornará meu melhor amigo se fizer isto!

Entrada:
As duas primeiras linhas conterão, cada uma, uma lista de códigos de produtos, sendo que os códigos estarão sempre separados por um único espaço em branco e o último código será sempre igual a -1 (menos um), ou seja, ele é um *flag* para indicar o final daquela lista.

Saída:
Se as listas forem *iguais*, a saída deverá ter uma única linha contendo, nesta ordem, a palavra `iguais` (grafada em letras minúsculas), um espaço em branco, e o número de códigos iguais.
Se as listas forem *diferentes*, a saída deverá ter uma única linha contendo, nesta ordem, a palavra `diferentes` (grafada em letras minúsculas), um espaço em branco e, por fim, a sequência de códigos distintos que diferenciam as listas. Os códigos devem ser apresentados em ordem crescente, e sem repetição, sempre separados por um único espaço em branco entre eles.

Exemplo 01:

ENTRADA	SAÍDA
1 2 3 2 5 6 5 8 7 16 7 20 9 13 29 -1	iguais 12
20 29 5 7 9 1 5 2 7 13 3 16 6 8 2 -1	

Exemplo 02:

ENTRADA	SAÍDA
1 4 16 7 14 2 15 9 3 10 16 -1	diferentes 1 2 3 4 7 14
2 16 10 10 16 15 9 2 16 15 -1	

Exemplo 03:

ENTRADA	SAÍDA
1 2 2 1 1 1 2 3 -1	iguais 3
1 2 3 -1	

Exemplo 04:

ENTRADA	SAÍDA
1265 5684 125 333 125 111 469 737 1652 16798 123 737 111 333 -1	iguais 10
5684 737 1652 123 111 469 333 1265 16798 125 -1 -1	

Observação: Cada uma das listas pode ter até uma sequência de até 1000 códigos, sendo que cada código de produto está no intervalo de 1 a 50000, inclusive extremos.

Questão 02 (5,0 pontos) Uma operação comum em diversas áreas da computação científica é a adição de números inteiros positivos com grande número de dígitos. Por exemplo, somar dois números de 40 dígitos cada, o que pode gerar um número de até 41 dígitos.
Você está participando de uma equipe de desenvolvimento de uma aplicação científica que deve implementar a operação de adição mencionada.
A aplicação deve ser desenvolvida utilizando a linguagem ℂ, conforme a seguir especificado.

Entrada
A primeira linha da entrada conterá o número de casos de teste, t, a serem aplicados. Sabe-se que $1 \leq t \leq 50$.
A seguir são apresentadas t linhas, cada uma contendo os dois números inteiros a serem adicionados, digamos m e n, sabendo-se que eles terão no máximo 40 dígitos cada, mas que também poderão ser iguais a 0 (zero). A dupla de números está separada por um único espaço em branco.

Saída
A saída consiste de t linhas, cada uma com o resultado da operação de adição dos pares de números correspondentes, na ordem em que foram fornecidos.

Exemplo 01:

ENTRADA	SAÍDA
1	9547748898511
9423891297239 123857601272	

Exemplo 02:

ENTRADA	SAÍDA
2	9547748898511
9423891297239 123857601272	737238114780433811713598
737238112845712940348123 1934720871365475	

Exemplo 03:

ENTRADA	SAÍDA
3	104759
0 104759	105331
105331 0	0
0 0	

Exemplo 04:

ENTRADA	SAÍDA
1	19999999999999999999999999999999999998
9..9 (39 dígitos 9) 9..9 (39 dígitos 9)	

Questão 03 (5,0 pontos). Num programa ℂ foi declarado o seguinte TAD (Tipo Abstrato de Dados):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define SUCESSO 1
5  #define FALHA -1
6
7  typedef struct Celula* ApontadorCelula;
8
9  typedef struct Celula {
10     ApontadorCelula ant;
11     char dado;
12     ApontadorCelula prox;
13 } Celula;
```

O objetivo é utilizá-lo para construir uma aplicação que seja capaz de manipular cadeias de caracteres de tamanho extremamente grandes, ou seja, que podem conter bilhões de caracteres, pois estas cadeias representarão porções de ADN (*Ácido Desoxirribonucleico*). Para isso cada cadeia somente poderá conter os seguintes caracteres, sempre em grafados em maiúsculo:

A – correspondente à base nitrogenada *adenina*;

C – *citosina*;

G – *guanina*;

T – *timina*.

Você deverá implementar um programa, em ℂ, que seja capaz receber como entrada uma sequência de caracteres, representando um trecho de certo ADN, e armazená-la numa instância desta TAD. Vamos chamá-la de ADN_{org}.
Em sequência, deve ser gerada uma outra instância desta TAD para armazenar a "sequência ADN complementar de ADN_{org}", onde a base A (adenina) é substituída pela base T (timina) e a base C (citosina) é substuída pela base G (guanina), como também o contrário em ambos casos. Vamos chamar esta instância de ADN_{cmp}.

Por fim, deve-se imprimir ADN_{cmp}, um único espaço em branco, e a quantidade de bases nitrogenadas que ela possui.

Entrada
A única linha da entrada contém a sequência de caracteres que representa um trecho de ADN e, finalizando, o símbolo $ (cifrão), utilizando como *flag* indicado de fim de entrada.

Saída
A única linha da saída contém a correspondente "sequência ADN complementar" da sequência de entrada e SEM espaço em branco entre os caracteres e, por fim, o numeral que representa a quantidade total de bases nitrogenadas que ADN_{cmp} possui.

Exemplo 01:

ENTRADA	SAÍDA
ACGTAACCTTACG$	TGCATTGGAATGC 13

Exemplo 02:

ENTRADA	SAÍDA
ACGT$	TGCA 4

Observações:

(a) A sequência de entrada pode ter, para efeito de teste, até 1000 caracteres úteis, ou seja, A, C, G ou T, sempre grafados em letras maiúsculas;

(b) A resolução somente será validada se utilizar, para a representação de ADN_{org} e ADN_{cmp} o TAD declarado, ou seja, a utilização de quaisquer outras estruturas de dados implicará na atribuição da nota $0,0$ (zero) à questão, mesmo que todos os "casos de teste" corrigidos pelo *Sharif Judge System* do INF/UFG sejam assinados como "corretos".

(c) Note que, na saída, não há espaço entre os caracteres impressos.

Questão 04 (5,0 pontos). Você faz parte da equipe de desenvolvimento de *softwares* da empresa *Gauss Software Inc.* e, nesta condição, ficou responsável por implementar – utilizando a linguagem de programação $\mathbb{C}$ – uma aplicação para manipular uma fila de prioridades, `FP`, que gerencia, num sistema operacional multitarefa (SOM), a impressão de arquivos.

As seguintes definições/regras são aplicáveis à fila `FP` de arquivos para impressão:

(1) Cada arquivo a ser impresso pelo SOM é identificado por dois números naturais: (1º) o código do arquivo, C_a, e a prioridade para sua impressão, P_a. O valores de C_a são únicos, ou seja, se constituem numa chave primária.

(2) A fila `FP` está inicialmente vazia e recebe, como entrada, pares $C_a - P_a$. Cada par representa um arquivo a ser impresso, sendo os números do par separados por um único espaço em branco entre eles.

(3) A impressão é realizada de acordo com a prioridade: arquivos com códigos P_a menores (alta prioridade) são impressos antes de arquivos com códigos maiores (baixa prioridade). Havendo empate entre dois arquivos, deve ser primeiramente impresso aquele que tiver menor valor para C_a.

Você deverá implementar um programa que seja capaz de imprimir, numa única linha de saída, os códigos dos arquivos na ordem em que estes serão impressos, da esquerda para a direita, pelo SOM.

Entrada
A primeira linha da entrada contém uma sequência de pares $C_a - P_a$, sendo os elementos sempre separados por um único espaço em branco entre eles, como também em relação ao par seguinte (se houver). O último par, que será utilizado como *flag* para detecção de fim de entrada, é sempre $-1 - 1$ (menos um, espaço em branco, menos um).

Observações: Considere que:

- $1 \leq C_a \leq 1000$;
- $1 \leq P_a \leq 100$;
- A linha de entrada pode ter até 1000 pares $C_a - P_a$, além do par $-1 - 1$.

Saída
Os códigos dos arquivos na ordem em que serão impressos, ou seja: código do primeiro arquivo a ser impresso seguido do código do segundo arquivo a ser impresso e, assim, sucessivamente. Os códigos são separados por um único espaço em branco.

Exemplo 01:

ENTRADA	SAÍDA
4 70 6 30 2 30 8 50 1 10 5 100 -1 -1	1 2 6 8 4 5

Exemplo 02:

ENTRADA	SAÍDA
76 100 37 70 40 50 58 100 57 50 32 70 -1 -1	40 57 32 37 58 76

Exemplo 03:

ENTRADA	SAÍDA
55 30 44 30 37 30 26 30 81 30 92 30 25 30 -1 -1	25 26 37 44 55 81 92

Exemplo 04:

ENTRADA	SAÍDA
44 10 -1 -1	44