Fundação CECIERJ - **Vice-Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AD2 – 2° semestre de 2017

IMPORTANTE

- As respostas (programas) deverão ser entregues pela plataforma em um arquivo ZIP contendo todos os arquivos de código fonte (extensão ".py") necessários para que os programas sejam testados. Respostas entregues fora do formato especificado, por exemplo, em arquivos com extensão ".pdf", ".doc" ou outras, não serão corrigidas.
- Serão aceitos apenas soluções escritas na linguagem Python 3. Programas com erro de intepretação não serão corrigidos. Evite problemas utilizando tanto a versão da linguagem de programação (Python 3.X) quanto a IDE (PyCharm) indicadas na Aula 1.
- Quando o enunciado de uma questão inclui especificação de formato de entrada e saída, tal especificação deve ser seguida à risca pelo programa entregue. Atender ao enunciado faz parte da avaliação e da composição da nota final.
- Faça uso de boas práticas de programação, em especial, na escolha de identificadores de variáveis, subprogramas e comentários no código.
- As respostas deverão ser entregues pela atividade "Entrega de AD2" antes da data final de entrega estabelecida no calendário de entrega de ADs. Não serão aceitas entregas tardias ou substituição de respostas após término do prazo.
- As ADs são um mecanismo de avaliação individual. As soluções podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual. Respostas plagiadas não serão corrigidas.

1ª Questão (2,0 pontos)

Faça um programa que represente dois polinômios, p(x) e q(x), definidos pelo usuário.

Produza e escreva a soma dos dois polinômios lidos, onde soma(x) = p(x)+q(x), e a multiplicação dos dois polinômios lidos, onde multiplica(x) = p(x)*q(x). Suponha que os polinômios tenham coeficientes que sejam números em ponto flutuante e os graus sejam inteiros, o coeficiente de grau maior deve ser lido antes.

Entrada

Na primeira linha devem ser escritos todos os coeficientes do primeiro polinômio, separados por espaços em branco. Na segunda linha devem ser escritos todos os coeficientes do segundo polinômio, separados por espaços em branco.

Saída

Na primeira linha deve ser escrita a string “Coeficientes do Polinômio Soma: ”, seguida dos coeficientes calculados;

Na segunda linha deve ser escrita a string “Coeficientes do Polinômio Multiplicação:”, seguida dos respectivos coeficientes calculados.

Dica

Represente cada polinômio como um vetor, onde os coeficientes de maior grau estejam mais à esquerda.

Exemplo

Entrada
3.7 1.8 -2 1.0 1.0

Saída
Coeficientes do Polinômio Soma: 3.7 2.8 -1.0 Coeficientes do Polinômio Multiplicação: 3.7 5.5 -0.2 -2.0

Distribuição de Pontos

Leitura da entrada - 10%; Uso de subprogramação - 20%; Representação correta do polinômio no vetor - 20%; Escrita dos coeficientes do polinômio soma - 20%.; Escrita dos coeficientes do polinômio multiplicação - 30%

2ª Questão (2,0 pontos)

A prefeitura da sua cidade resolveu lhe contratar para ajudar aos habitantes a localizarem qual a farmácia mais próxima, através de um programa que processe arquivos contendo o posicionamento de todas as farmácias da sua cidade. Seu programa deve dizer qual o posicionamento da farmácia mais próxima de um cidadão que execute o seu programa. Caso haja empate, ele deve dizer o posicionamento de uma delas. O arquivo texto de nome “farmacias.txt”, fornecido pela prefeitura, contém o posicionamento de todas as farmácias, onde cada linha possui dois números inteiros.

Entrada

Seu programa deve ler o posicionamento, isto é, as coordenadas XCidadao, YCidadao, de um cidadão.

Saída

Seu programa deve escrever a mensagem "Arquivo de Farmácias está vazio!!!", caso o arquivo lido esteja vazio. Caso contrário, deve escrever na primeira linha a string “Local da Farmácia Mais Próxima: ”, seguida do posicionamento XFarmacia, YFamacia que seja a mais próxima do cidadão, e a que distância ela se encontra. Na segunda linha deve escrever a string “Distância à Percorrer: ”, seguida da menor distância encontrada.

Dica

A distância entre dois pontos, (X1, Y1) e (X2, Y2), é dada por $\text{sqrt}((X1 - X2)^2 + (Y1 - Y2)^2)$.

Exemplos

Entrada
Arquivo “farmacias.txt”
300 200 10 56 25 41 90 18 70 90 100 50
Entrada padrão
20 88

Saída
Local da Farmácia Mais Próxima: 10 56 Distância à Percorrer: 33.52610922848042

Distribuição de Pontos

Leitura da entrada - 10%; Leitura correta do arquivo - 20%; Escrita correta no caso de arquivo vazio - 20%; Escrita do local da farmácia mais próxima - 25%; Escrita da menor distância - 25%

3ª Questão (2,0 pontos)

O Campeonato de Futebol de Botão do Cederj é um evento bastante popular e você é uma peça fundamental na divulgação do andamento dos times. Dados os resultados dos jogos, sua tarefa é montar a tabela de classificação correspondente.

A tabela de classificação mostra a colocação de cada time em ordem decrescente de pontos, seguido pelo saldo de gols e o número de gols marcados. Quando existe empate no número de pontos, no saldo de gols e no número de gols marcados, você deve considerar que esses times ocupam a mesma posição na tabela de classificação e mostra-los em ordem alfabética.

O número de pontos de um time é dado pela soma de pontos obtidos em cada partida disputada por esse time, onde uma vitória lhe garante 3 (três) pontos, o empate resulta em 1 (um) ponto e a derrota em 0 (zero) pontos.

O saldo de gols de um time é dado pela quantidade de gols marcados menos a quantidade de gols sofridos em todas as partidas disputadas por esse time.

Entrada

A entrada é o arquivo texto de nome “partidas.txt”, fornecido pelo dirigente do Campeonato de Futebol de Botão do Cederj ao término de cada rodada da competição.

A primeira linha do arquivo contém dois valores inteiros T ($1 \leq T \leq 99$) e J ($1 \leq J \leq 10$), onde T é o número de times e J é o número de jogos disputados.

Cada uma das T linhas seguintes contem o nome de um time. É garantido que os nomes dos times sejam todos diferentes e compostos por uma única palavra cada. Nenhum nome de time tem mais que 20 caracteres.

Por fim, as J linhas seguintes contém o resultado de cada jogo. Os jogos são mostrados no seguinte formato: nome do primeiro time, espaço em branco, número de gols marcados pelo

primeiro time, um espaço em branco, a letra “x”, um espaço em branco, número de gols marcados pelo segundo time, um espaço em branco e o nome do segundo time.

É garantido que T e J atenderão os limites indicados. Logo, não é preciso fazer a verificação.

Saída

Seu programa deve emitir o arquivo “tabela.txt”, contendo a tabela de classificação conforme o formato e alinhamento apresentado no exemplo. Nessa tabela, os times aparecem em ordem de classificação, um por linha. Cada linha é composta pela posição do time no campeonato (dois caracteres), nome do time (20 caracteres), número de pontos (dois caracteres), número de jogos disputados (dois caracteres), número de gols marcados (dois caracteres), número de gols sofridos (dois caracteres) e saldo de gols (três caracteres). Note que se vários times estão empatados, todos recebem a mesma posição e são apresentados em ordem alfabética. Deve ser colocado um espaço em branco entre cada duas colunas vizinhas da tabela.

Exemplo

Entrada (paridas.txt)

```
4 3
OsBotões
LeoNoFutebol
Cacareco
ClubeDoJuca
OsBotões 6 x 0 Cacareco
Cacareco 0 x 2 ClubeDoJuca
Cacareco 0 x 2 LeoNoFutebol
```

Saída (tabela.txt)

```
 1              OsBotões  3  1  6  0   6
 2          ClubeDoJuca  3  1  2  0   2
 2         LeoNoFutebol  3  1  2  0   2
 3              Cacareco  0  3  0 10 -10
```

Dica

Utilize o operador de formação (%) para tabular corretamente a saída. Em especial, utilize a máscara de formatação “%*n*d” na formatação de valores inteiros, onde *n* é o número de caracteres que se deseja imprimir na colunas da tabela de classificação, sendo que caracteres à esquerda do número são preenchidos com espaços em branco. Por exemplo, `"%5d" % 10` retorna a string `"    10"`, de comprimento 5. Para o caso de texto, a máscara “%*n*s” emite uma string com *n* caracteres, sendo que o texto informado é alinhado à direita. Por exemplo `"%7s" % "teste"` retorna a string `"  teste"`, de comprimento 7.

Restrição

O processo de ordenação dos times na tabela de classificação deve ser implementado pelo aluno. Portanto, é proibido o uso de funções de ordenação prontas disponíveis na linguagem Python. Será atribuída nota 0 (zero) para soluções que façam uso de funções de ordenação prontas, mesmo que o resultado final esteja correto.

Distribuição de Pontos

Leitura da entrada - 20%; Montagem da tabela de classificação ordenada - 50%; Escrita da saída - 30%

4ª Questão (2,0 pontos)

Juca é Coordenador de Qualidade do Cederj é frequentemente visita os polos para ver como andam as coisas. Isso faz com que Juca viaje por todo o estado do Rio de Janeiro (uma tarefa bastante cansativa), passando vários dias longe de casa. Ao chegar em casa depois de uma semana cansativa, Juca percebeu que estava sem suas anotações. Logo deduziu que as havia esquecido em um dos polos visitados por ele, então decidiu voltar para busca-las.

Após procurar em todos os polos em que esteve nos últimos três dias, Juca ainda não encontrou suas anotações. Então resolveu pedir sua ajuda para procurar novamente. Para isso ele informará alguns dos polos onde ele esteve na última semana. Ajude Juca a encontrar as anotações informando em quais polos é possível que ele tenha esquecido o material.

Entrada

A entrada é um arquivo binário de nome “entrada.bin”. O arquivo é sequencial e composto por dois valores inteiros (ocupando 4 bytes cada), Q ($1 \leq Q \leq 1000$) e E ($1 \leq E \leq Q$) representado, respectivamente, a quantidade de polos onde Juca esteve na última semana e a quantidade de polos onde esteve nos últimos três dias.

Em seguida são apresentados E inteiros S_i ($1 \leq S_i \leq 1000$) contendo o número de identificação de cada um dos polos em que ele esteve nos últimos três dias (cada inteiro S_i ocupa 4 bytes). Seguem Q inteiros C_i ($1 \leq C_i \leq 1000$) contendo o número de identificação de cada um dos polos em que Juca esteve durante a última semana (cada inteiro C_i ocupa 4 bytes).

É garantido que Q, E, S_i e C_i atenderão os limites indicados. Logo, não é preciso fazer a verificação.

Saída

Seu programa deverá gerar um arquivo binário de nome “saida.bin” contendo o resultado. Para cada polo em que Juca esteve na última semana seu programa deverá coloca sequencialmente no arquivo o valor inteiro 0 (ocupando 4 bytes) caso ele já tenha visitado esse polo ao procurar pelas anotações, ou o valor inteiro 1 (ocupando 4 bytes) caso ele não tenha visitado esse polo ainda enquanto procurava pelo material.

Exemplo

O exemplo que segue montra o conteúdo dos arquivos binários de entrada e saída de forma textual apenas para facilitar a leitura dos dados. Você deve estar ciente que esses arquivos são na prática binários, tendo cada valor armazenado como um valor numérico inteiro com 4 bytes.

Entrada (entrada.bin)
10 5 1 15 5 998 27 1 88 15 88 99 5 100 7 27 998

Saída (saida.bin)
0 1 0 0 1 0 1 1 0 0

Dica

Para fins de depuração de código, escreva um programa auxiliar que crie arquivos binários contendo nada mais do que valores armazenados sequencialmente conforme a especificação

do enunciado. Escreva, também, programas auxiliares que ajudem a verificar se o conteúdo do arquivo de saída está correto. Os programas auxiliares não devem ser entregues junto com a solução da questão. Caso sejam entregues, os mesmos não serão considerados na correção, independentemente de estarem corretos ou errados.

Restrição

Se a questão for resolvida considerando arquivos texto então a nota atribuída para a mesma será 0 (zero), mesmo que a solução esteja correta no contexto de arquivos texto.

Distribuição de Pontos

Leitura da entrada - 25%; Uso de conjunto (`set`) na solução - 50%; Escrita da saída - 25%

5ª Questão (2,0 pontos)

O controle de frequência nas Avaliações Presenciais (APs) do Cederj é feito por listas de presença assinadas pelos alunos de cada turma. Para verificar se de fato são os alunos que comparecem às provas, é necessário que seja feita a conferência das assinaturas. O problema é que existem muitos alunos em cada turma, de modo que a conferência manual é inviável.

Você foi contratado para implementar um sistema que, dada a coleção de assinaturas cadastradas por turma, verifique se as assinaturas na lista de presença da AP da disciplina são válidas ou são faltas. O sistema deve ser capaz de tolerar pequenas variações. Afinal, duas assinaturas de uma mesma pessoa nem sempre são idênticas.

Uma assinatura é considerada falsa se houver mais de uma diferença entre a original e a que estiver sendo checada. Considere diferença uma troca de maiúscula para minúscula ou o contrário.

Entrada

Todas as listas de presença de um dia de APs foram colocadas no arquivo texto "listas_de_presenca.txt". Logo, haverá diversas turmas cujas assinaturas deverão ser verificas (veja o exemplo).

A primeira linha de cada turma indica o nome da disciplina.

A segunda linha de cada turma inicia com um inteiros N ($1 \leq N \leq 50$) representando a quantidade de alunos na turma. As próximas N linhas serão da seguinte forma: nome do aluno composto por uma única palavra, um espaço em branco, e assinatura original composta por uma palavra. É garantido que nenhum nome se repita em uma dada turma.

A seguir haverá um inteiro M ($0 \leq M \leq N$), representando a quantidade de alunos que compareceram à AP da turma atual. M linhas seguem, no seguinte formato: nome do aluno composto por uma única palavra, um espaço em branco, e assinatura posta na lista de presenças. É garantido que nenhum nome se repita em uma dada lista de presença e que todos os nomes façam parte da lista de alunos da turma atual.

A entrada termina com o nome de disciplina igual a "Fim".

É garantido que N e M atenderão os limites indicados. Logo, não é preciso fazer a verificação.

Saída

Seu programa deve produzir o arquivo "quantidade_de_falsificacoes.txt" (veja o exemplo). Esse arquivo deve conter uma linha para cada turma. Essa linha deve ser composta pelo nome da

disciplina, seguido por ":" e um espaço em branco, e a quantidade de assinaturas falsas encontradas.

Exemplo

Entrada (listas_de_presenca.txt)	Saída (quantidade_de_falsificacoes.txt)
Fundamentos de Programação 4 Ana anA Pedro PeDRO Maria Maria Paulo PaulO 3 Paulo PaulO Ana anA Pedro PEdro Fundamentos de Algoritmos 2 Andre Andre Carolina CaRoLiNa 2 Carolina CAROLINA Andre AnDRE Fim	Fundamentos de Programação: 1 Fundamentos de Algoritmos: 2

Distribuição de Pontos

Leitura da entrada - 20%; Uso de dicionário (`dict`) na solução - 60%; Escrita da saída - 20%

Boa Avaliação!