Dredd - Juiz Online

Principal | Perfil | Minhas Provas | Sair

Minutos Restantes: 1461

Usuário: Gabriel Nathan Almeida Silva

Notas:
Q1: ?
Q2: 90.3
Q3: ?
Q4: ?
Q5: ?
Q6: ?
Q7: 100
Total: 27

REO 2 - Lista, Deque, Sequence Set e Hash

Prova Aberta Até: 28/06/2020 23:59:59

Número Máximo de Tentativas: 4

Atenuação da Nota por Tentativa: 0%

Instruções para a prova: A prova é individual. Desligue seu celular. Não converse com os colegas. Não fique olhando para a tela dos colegas.

Questão 1: Concatena listas removendo repetidos

Implemente uma função que receba duas listas encadeadas de valores inteiros positivos e transfira para o final da primeira lista os elementos da segunda que não estejam presentes na primeira. Além disso, os elementos presentes em ambas as listas serão removidos da primeira lista também. No final, a primeira lista representará a concatenação das duas listas sem os elementos que aparecem em ambas as listas.

Obs: Considere que não existem entradas repetidas em uma mesma lista.

Entradas:

1. Quantidade de elementos a serem inseridos na primeira lista.
2. Elementos da primeira lista (sem repetição de elementos).
3. Quantidade de elementos a serem inseridos na segunda lista.
4. Elementos da segunda lista (sem repetição de elementos).

Saídas:

1. Elementos da lista 1 seguidos pelos elementos da lista 2 que não estão presentes na primeira.

Exemplo de Entrada:

```
4
2 6 0 3
7
11 8 3 4 1 56 6
```

Exemplo de Saída:

```
2 0 11 8 4 1 56
```

Exemplo de Entrada:

```
5
1 8 21 15 0
7
11 8 3 4 21 56 15
```

Exemplo de Saída:

```
1 0 11 3 4 56
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo | Nenhum arquivo selecionado | Enviar Resposta

Questão 2: Criar lista duplamente encadeada com método inverte()

Você deve implementar uma lista duplamente encadeada e adicionar o método **inverte()**, que inverte a ordem dos elementos da lista sem modificar o conteúdo de cada nó, ou seja, você deve alterar somente as referências `proximo` e `anterior` dos nós.

Entradas:

Uma sequência de caracteres e palavras tal que:

- I: Insere uma palavra na lista original
- X: Executa a função inverte
- P: Imprime a lista em ordem direta
- R: Imprime a lista em ordem reversa
- Q: Encerra os comandos

Saídas:

1. Impressões da lista pelos comandos
2. Impressão da lista em ordem direta e reversa após o comando Q

Exemplo de Entrada:

```
I Horacio I Sales I Beltrao P X P I GCC216 Q
```

Exemplo de Saída:

```
Horacio Sales Beltrao
Beltrao Sales Horacio
Beltrao Sales Horacio GCC216
GCC216 Horacio Sales Beltrao
```

Exemplo de Entrada:

```
I 1 I 2 I 3 I 4 I 5 I 6
X
P
R
X
P
R
I 7 I 8 I 9
X
R
I 10 I 11 I 12
X
P
I 13
Q
```

Exemplo de Saída:

```
6 5 4 3 2 1
1 2 3 4 5 6
1 2 3 4 5 6
6 5 4 3 2 1
1 2 3 4 5 6 7 8 9
12 11 10 1 2 3 4 5 6 7 8 9
12 11 10 1 2 3 4 5 6 7 8 9 13
13 9 8 7 6 5 4 3 2 1 10 11 12
```

Peso: 1

Última tentativa realizada em: 27/06/2020 23:29:12

Tentativas: 1 de 4

Nota (0 a 100): 90.3

Status ou Justificativa de Nota: A quantidade de dados escritos pelo programa é diferente da quantidade de dados esperados.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo Nenhum arquivo selecionado Enviar Resposta

Questão 3: Criar lista com método para simular união de conjuntos

Implemente uma lista simplesmente encadeada que represente um conjunto, ou seja, suas listas não devem aceitar elementos repetidos. Sendo assim, implemente um método que receba como parâmetro uma outra lista para realizar a união, de modo que o objeto que chamou o método seja aquele que vai representar a união. **Dica:** A = {1, 2, 3, 4} e B = {2, 4, 6, 8}, A U B = {1, 2, 3, 4, 6, 8}.

Entradas:

1. Número de elementos do primeiro conjunto.
2. Elementos do primeiro conjunto.
3. Número de elementos do segundo conjunto.
4. Elementos do segundo conjunto.

Saídas:

1. Elementos do primeiro conjunto.
2. Elementos do segundo conjunto.
3. União dos dois conjuntos.

Exemplo de Entrada:

```
4
1 3 5 7
4
2 4 6 8
```

Exemplo de Saída:

```
1 3 5 7
2 4 6 8
1 3 5 7 2 4 6 8
```

Exemplo de Entrada:

```
3
1 2 3
3
3 30 300
```

Exemplo de Saída:

```
1 2 3
3 30 300
1 2 3 30 300
```

Exemplo de Entrada:

```
0
4
2 2 2 2
```

Exemplo de Saída:

```
2
2
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo Nenhum arquivo selecionado Enviar Resposta

Questão 4: Deque - controle de undo

Vários programas tem um comando para desfazer a operação anterior, conhecido como *undo*. Usar este comando seguidamente faz com que a última ação seja desfeita, depois a penúltima e assim por diante.

Para não gastar muita memória é comum que haja um limite de ações disponíveis para o *undo*. Por exemplo, pode ser possível desfazer apenas as últimas 20 operações realizadas.

Este tipo de funcionalidade é mais fácil de controlar usando uma estrutura do tipo *deque* (*double ended queue*). Use alguma implementação de fila ou de lista para criar uma classe `Deque`. Depois, faça um programa para guardar e retirar linhas de texto, que representam comandos.

Entradas:

Inicialmente seu programa deve ler um número natural que representa a quantidade máxima de instruções que a estrutura de dados vai guardar. Depois várias linhas de texto serão lidas. Sempre que a linha for "undo", o programa deve dizer que desfez a última instrução. Se a instrução não for "undo", o programa deve guardar a linha na lista de coisas a desfazer. Se a capacidade máxima for atingida, o programa deve "esquecer" a instrução mais antiga antes de inserir a nova instrução na estrutura. O comando especial "sair" faz com que a execução termine.

1. Capacidade da fila de comandos;
2. comandos, cada um numa linha.

Saídas:

O comando "undo" deve produzir a saída: `desfazer: comando que foi desfeito`. Toda vez que a capacidade máxima for atingida, o programa deve produzir a saída: esqueci: comando que foi removido. No caso de tentarem desfazer ação que não existe, o programa deve escrever ERRO (letras maiúsculas) na saída padrão.

Exemplo de Entrada:

```
3
comando um
comando dois
undo
comando três
comando quatro
comando cinco
sair
```

Exemplo de Saída:

```
desfazer: comando dois
esqueci: comando um
```


Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo | Nenhum arquivo selecionado | Enviar Resposta

Questão 5: Criar hash com método para simular conjuntos

Uma estrutura de dados do tipo conjunto (*set*) é uma coleção de elementos. Dado os conjuntos *S*, *A* e *B*, e um elemento *x*, um conjunto geralmente implementa as seguintes operações básicas:

- `membro(S,x)` – Retorna true se o item *x* for membro do conjunto *S*.
- `adiciona(S,x)` – Adiciona o elemento *x* no conjunto *S*, descartando duplicatas. Se *x* já fizer parte de *S*, não é gerado erro, apenas não se modifica o conteúdo de *S*.
- `remove(S,x)` – Remove o elemento *x* do conjunto *S*. Se *x* não fizer parte de *S*, não é gerado erro, apenas não se modifica o conteúdo de *S*.
- `tamanho(S)` – Retorna o número de elementos de S.
- `imprime(S)` – Imprime os elementos de S. Caso o conjunto seja vazio, imprime {}
- `uniao(A,B)` – Retorna um novo conjunto $A \cup B$, formado pela união dos elementos de *A* e *B*.
- `intersecao(A,B)` – Retorna um novo conjunto $A \cap B$, formado pela interseção dos elementos de *A* e *B*.
- `diferenca(A,B)` – Retorna um novo conjunto *A* - *B*, formado pelos elementos de *A* que não fazem parte de *B*.

Como pode-se perceber, um conjunto é muito similar a uma lista, com algumas particularidades (a impossibilidade de duplicatas, por exemplo). Conjuntos podem ser implementados de várias formas, incluindo: i) dicionários (tabelas hash); ii) árvores ou iii) listas dinâmicas com ponteiros. No caso de listas dinâmicas, os dados podem ainda ser armazenados por meio de ponteiros ou arranjos.

Neste exercício, você deverá implementar a hash e os métodos para simular as operações básicas de conjuntos listadas anteriormente. Perceba que as operações de `uniao(A,B)`, `intersecao(A,B)` e `diferenca(A,B)` não precisam ser necessariamente implementadas, uma vez que estamos utilizando dicionários (**tabelas hash**).

Para este problema em específico, o conjunto estará limitado a 100 elementos inteiros, portanto, utilize a função de espelhamento dada por `hash(k) = k mod 101`. O vetor de armazenamento do hash terá obviamente 101 posições. O valor associado à chave (o elemento) a ser armazenado é 1 (para indicar pertinência do elemento ao conjunto) ou -1 (quando o elemento não pertence ao conjunto). Obviamente, como o conjunto inicialmente encontra-se vazio, todas os valores nas posições do vetor de armazenamento devem ser inicializadas em -1. Obviamente também, por usar um vetor para armazenamento, **o tratamento de colisões deverá ser por endereçamento aberto.**

Além da estrutura de dados, você deverá implementar uma aplicação básica para teste, que irá criar um conjunto e executará uma série de operações na seguinte ordem:

1. Constrói o conjunto a partir de uma lista de 10 elementos digitada pelo usuário.
2. Remove três elementos indicados pelo usuário.
3. Verifica se um dado valor informado pelo usuário é membro do conjunto (imprima 1 para pertinência e -1 para não-pertinência.
4. Repete a operação de verificação de pertinência para outro valor informado pelo usuário.
5. Imprime o conteúdo do conjunto, na ordem em que estão armazenados.
6. Adiciona três novos elementos indicados pelo usuário
7. Imprime novamente o conteúdo do conjunto, na ordem em que estão armazenados.

Entradas:

1. 10 elementos a serem inseridos no conjunto
2. 3 elementos a serem removidos do conjunto
3. 2 valores para verificação de pertinência
4. 3 elementos a serem inseridos no conjunto

Saídas:

1. Informação sobre pertinência de um valor fornecido pelo usuário
2. Informação sobre pertinência de outro valor fornecido pelo usuário
3. Elementos do conjunto
4. Elementos do conjunto, após inserção de novos elementos

Exemplo de Entrada:

```
1 2 3 4 5 6 7 8 9 10
9 10 11
7 10
1 9 10
```

Exemplo de Saída:

```
1
-1
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9 10
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo Nenhum arquivo selecionado Enviar Resposta

Questão 6: Tabela hash - Dicionário

Utilize uma tabela hash para implementar um dicionário de capacidade para 23 palavras, sua tabela terá o tratamento de colisões por encadeamento. Seu programa deve ser capaz de armazenar e buscar por uma palavra e seu significado. Caso seja feita uma busca por uma palavra inexistente em seu dicionário, o programa imprimirá NULL no lugar do significado da palavra. Seu programa deve permitir quantas buscas forem desejadas pelo usuário, sendo -1 a palavra que representa a intenção de termino do programa. **OBS:** Sua função que faz o hash será da seguinte forma: tamanho_da_palavra mod 23.

Entradas:

1. Quantidade de palavras a serem inseridas.
2. Palavra.
3. Significado da Palavra.
4. Palavras buscadas.

Saídas:

1. [Palavra_buscada] => Significado da palavra buscada.

Exemplo de Entrada:

```
5
Casa Edifício de formatos e tamanhos variados, ger. de um ou dois andares, quase sempre destinado à habitaçã
Aniversário Diz-se de ou dia em que se completa um ou mais anos em que se deu determinado acontecimento.
Carta Mensagem, manuscrita ou impressa, a uma pessoa ou a uma organização, para comunicar-lhe algo.
Exonerado  Libertar ou libertar-se de uma obrigação ou de um dever.
Concomitantemente Que acompanha ou coexiste. Que acontece ou se faz ao mesmo tempo.
Carta
Casa
Boneca
-1
```

Exemplo de Saída:

```
[Carta] => Mensagem, manuscrita ou impressa, a uma pessoa ou a uma organização, para comunicar-lhe algo.
[Casa] => Edifício de formatos e tamanhos variados, ger. de um ou dois andares, quase sempre destinado à hab
[Boneca] => NULL
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo Nenhum arquivo selecionado Enviar Resposta

Questão 7: Hash com endereçamento aberto e redimensionamento

Implemente uma tabela hash com tratamento de colisão por endereçamento aberto. Use o código fornecido. A implementação deve seguir a seguinte estratégia:

- A tabela hash é criada para uma determinada capacidade. Toda vez que a inserção encontrar estrutura cheia, a capacidade aumenta de forma a permitir o armazenamento de cinco novos elementos.
- Os espaços de armazenamento podem estar em um de três estados; 1) VAZIO, 2) REMOVIDO, 3) COM VALOR. Todos os espaços começam VAZIOS e depois de alterados nunca mais voltam a ser VAZIOS.
- Em caso de colisão, usa-se a posição consecutiva e assim sucessivamente. A posição que sucede a última é a primeira.
- A busca de itens na tabela hash termina ao encontrar um espaço VAZIO.
- A tabela hash armazena pares chave/valor, onde a chave é do tipo texto e o valor é de tipo indeterminado. A chave vazia (string vazia) é especial e não pode ser associada a valores. Não é permitido criar outros valores especiais para chave, esperando que ninguém nunca use um determinado valor como chave.

Com relação ao tratamento de erros, o código deve ser a prova de programadores descuidados. Mandar remover de estrutura vazia, por exemplo, não pode passar despercebido, mas a decisão sobre o que fazer não pode ser tomada na classe Hash. A função main é a responsável por determinar o que será feito. Em todos os casos de erro, a função main vai simplesmente escrever "ERRO" na saída padrão. A implementação dada usa manipulação de exceções, mas você pode mudar para códigos de erros caso não tenha prática com manipulação de exceções.

A implementação dada usa ponteiros para determinar os 3 estados de uma posição do armazenamento o ponteiro especial REMOVIDO já está implementado. Atenção na hora de desalocar memória. É permitido alterar a forma como os 3 estados são implementados, mas você terá que alterar coisas que já estão prontas.

Existe um método pronto para escrever a posição, chave e valor de todos os valores da tabela hash. Esse método mostra posições VAZIAS, REMOVIDAS e ocupadas com sintaxes distintas. Se você quiser mudar a forma como os 3 estados são implementados deveria mudar este método sem alterar a saída que ele produz. A função hash que mapeia chaves (texto) em posições na tabela também está pronta e não pode ser alterada.

A recuperação do valor de uma chave deve ser eficiente: não basta encontrar um valor, é necessário atender a estratégia acima.

Entradas:

A parte de interface do programa já está implementada. Inicialmente é lido um valor para a capacidade da tabela hash. Em seguida, o programa recebe comandos e os executa. Cada comando produz uma saída específica. Os comandos são:

- A letra i, seguida de uma chave (palavra) e um valor (inteiro), para inserir uma chave/valor na estrutura.
- A letra r, seguida de uma chave (palavra) para remover uma chave/valor da estrutura.
- A letra c, seguida de uma chave (palavra) para consultar o valor de uma chave (o valor é escrito).
- A letra d, para debugar (escrever todas as informações armazenadas em formato específico.
- A letra f, para finalizar a execução do programa.

Exemplo de entrada e saída juntos:

```
3
i um 1
i dois 2
i tres 3
d
[0/dois/2] [1/tres/3] [2/um/1]
i quatro 4
d
[0] [1/um/1] [2] [3] [4] [5/tres/3] [6/dois/2] [7/quatro/4]
f
```

Exemplo de entrada e saída juntos:

```
3
i quatro 4
i tres 3
i dois 2
d
[0/quatro/4] [1/tres/3] [2/dois/2]
r dois
r tres
d
[0/quatro/4] [1/removido] [2/removido]
i um 1
d
[0/quatro/4] [1/removido] [2/um/1]
i cinco 5
d
[0/quatro/4] [1/cinco/5] [2/um/1]
i seis 6
d
[0] [1/um/1] [2] [3/cinco/5] [4] [5/quatro/4] [6/seis/6] [7]
f
```

Peso: 1

Última tentativa realizada em: 26/06/2020 22:48:06

Tentativas: 1 de 4

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher arquivo Nenhum arquivo selecionado Enviar Resposta

Minutos Restantes: 1461

Usuário: Gabriel Nathan Almeida Silva

Notas:
Q1: ?
Q2: 90.3
Q3: ?
Q4: ?
Q5: ?
Q6: ?
Q7: 100
Total: 27