Lista Extra de Exercícios

Questões sobre Pilha, Fila e Lista:

Questão 1: Faça um menu que dê ao usuário as seguintes opções:

(a) Se ele digitar 1 o programa deverá ler um inteiro e inserir o valor na fila (enqueue).

(b) Se ele digitar 2 o programa deverá retirar o elemento da fila e exibir o valor na tela (dequeue).

(c) Se ele digitar 3 o programa deverá exibir na tela o valor do elemento que está no início da fila(get_first).

(d) Se ele digitar 4 o programa deverá limpar a fila (clear).

(e) Se ele digitar 5 o programa deverá exibir a quantidade de elementos armazenados na fila (size).

(f) Se ele digitar 6 o programa deverá exibir todos os elementos da fila sem desempilhá-los (print).

(g) Se ele digitar 7 o programa deverá exibir todos os elementos da fila na ordem inversa.

(h) Se ele digitar 8 o programa deverá retirar apenas os elementos ímpares da fila.

(i) Se ele digitar 9 o programa deverá inverter os elementos da fila (inverter).

(j) Se ele digitar 10 o programa deverá finalizar.

(k) Se ele digitar qualquer outro valor, deverá ser ignorado.

(l) Se ele digitar 0, então o programa deverá finalizar; caso contrário, o programa voltará a execução no passo 1.

Questão 2: A conversão de números inteiros, na base 10, para outras bases numéricas se dá através de sucessivas divisões de um dado valor n pelo valor da base na qual se queira converter. Faça um programa para obter a conversão numérica, de acordo com a opção do usuário, utilizando a uma pilha:

(a) Decimal para Binario.
(b) Decimal para Octal.
(c) Decimal para Hexadecimal

Questão 3: Dada uma lista encadeada de caracteres formada por uma seqüência alternada de letras e dígitos, construa um método que retorne uma lista na qual as letras são mantidas na seqüência original e os dígitos são colocados na ordem inversa.

Exemplos:

A 1 E 5 T 7 W 8 G → A E T W G 8 7 5 1

3 C 9 H 4 Q 6 → C H Q 6 4 9 3

Como mostram os exemplos, as letras devem ser mostradas primeiro, seguidas dos dígitos.

Sugestões: - usar uma fila e uma pilha; - supor um método ehDigito() retorna booleano que retorna verdadeiro caso um caractere seja um dígito.

Questões sobre Árvore:

Questão 1: Crie um programa que guarde as informações de uma árvore genealógica de uma família. Os nomes dos membros devem ser guardados em um arquivo .txt

O seu programa deve ter as opções de:

+Adicionar novo membro da família

+Excluir membro

+Mudar o nome de membro’

+Procurar por membro

Questão 2: Crie um programa que registre os cadastros de clientes de 2 canis diferentes. Os dados devem ser cadastrados em 2 árvores distintas
O seu programa deve ter as opções de:

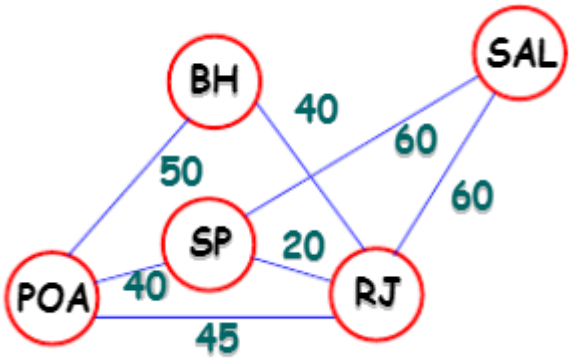
+ Cadastrar pet //usuário deve escolher em que canil efetuará cadastro.
+ Buscar pet //busca dve ser usando metodo do set
+ Ordenar pets

Questão 3: Crie um programa onde o usuário insira em uma árvore números aleatórios. O seu programa deve retornar o maior valor presente na árvore.

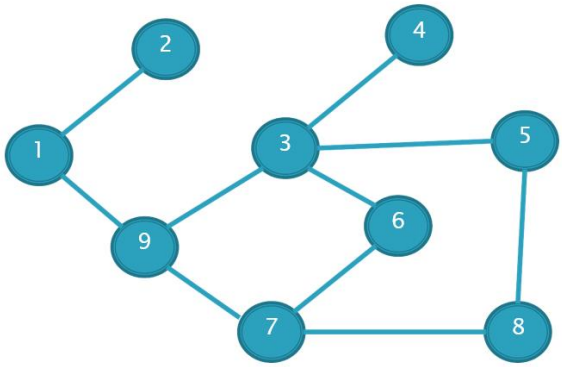
Questão 4: Crie um programa que retorne a média dos valores presentes numa árvore.

Questões sobre Grafos:

Questão 1: Dado o seguinte grafo, com suas respectivas arestas e vértices. Implemente um programa em C++ orientado a objetos, que retorne qual a rota mais barata de ir de SAL a POA.



Questão 2: Para o seguinte grafo, implemente uma função que realize busca em profundidade (DFS).



Questão 3: Para o mesmo grafo da questão 2, implemente agora uma função que realize busca em largura (BFS).

Questão 4: Tendo as questões implementadas, responda com suas palavras os seguintes questionamentos:

a) Qual as principais vantagens e desvantagens da utilização de grafos como estrutura de dados, dê exemplos concretos de suas principais aplicações.

b) Compare os tipos de busca em profundidade e largura, apresentado as vantagens e desvantagens de cada um dos tipos.

c) Cite as principais características e aplicações dos algoritmos para grafos, de Djikstra, Kruskal, Bellman-Ford e Ford-Fulkerson.

Questões sobre Hash:

Questão 1: As tabelas Hash, também conhecidas como tabelas de dispersão, armazenam elementos com base no valor absoluto de suas chaves e em técnicas de tratamento de colisões. Para o cálculo do endereço onde deve ser armazenada uma determinada chave, utiliza-se uma função denominada função de dispersão, que transforma a chave em um dos endereços disponíveis na tabela.

Suponha que uma aplicação utilize uma tabela de dispersão com 13 endereços-base (índices de 0 a 12) e empregue a função de dispersão h(x) = x mod 13, em que x representa a chave do elemento cujo endereço-base deve ser calculado.

Se a chave x for igual a 49, a função de dispersão retornará o valor 10, indicando o local onde esta chave deverá ser armazenada. Se a mesma aplicação considerar a inserção da chave 88, o cálculo retornará o mesmo valor 10, ocorrendo neste caso uma colisão. O Tratamento de colisões serve para resolver os conflitos nos casos onde mais de uma chave é mapeada para um mesmo endereço-base da tabela. Este tratamento pode considerar, ou o recálculo do endereço da chave ou o encadeamento externo ou exterior.

O professor gostaria então que você o auxiliasse com um programa que calcula o endereço para inserções de diversas chaves em algumas tabelas, com funções de dispersão e tratamento de colisão por encadeamento exterior.

Entrada

A entrada contém vários casos de teste. A primeira linha de entrada contém um inteiro N indicando a quantidade de casos de teste. Cada caso de teste é composto por duas linhas. A primeira linha contém um valor **M** ($1 \leq \mathbf{M} \leq 100$) que indica a quantidade de endereços-base na tabela (normalmente um número primo) seguido por um espaço e um valor **C** ($1 \leq \mathbf{C} \leq 200$) que indica a quantidade de chaves a serem armazenadas. A segunda linha contém cada uma das chaves (com valor entre 1 e 200), separadas por um espaço em branco.

Saída

A saída deverá ser impressa conforme os exemplos fornecidos abaixo, onde a quantidade de linhas de cada caso de teste é determinada pelo valor de **M**. Uma linha em branco deverá separar dois conjuntos de saída.

Questão 2: Você terá como uma entrada várias linhas, cada uma com uma string. O valor de cada caracter é computado como segue:

Valor = (Posição no alfabeto) + (Elemento de entrada) + (Posição do elemento)

Todas posições são baseadas em zero. 'A' tem posição 0 no alfabeto, 'B' tem posição 1 no alfabeto, ... O cálculo de hash retornado é a soma de todos os caracteres da entrada. Por exemplo, se a entrada for:
CBA
DDD

então cada caractere deverá ser computado como segue:

2 = 2 + 0 + 0 : 'C' no elemento 0 posição 0
2 = 1 + 0 + 1 : 'B' no elemento 0 posição 1
2 = 0 + 0 + 2 : 'A' no elemento 0 posição 2
4 = 3 + 1 + 0 : 'D' no elemento 1 posição 0
5 = 3 + 1 + 1 : 'D' no elemento 1 posição 1
6 = 3 + 1 + 2 : 'D' no elemento 1 posição 2

O cálculo final de hash será 2+2+2+4+5+6 = 21.

Entrada

A entrada contém vários casos de teste. A primeira linha de entrada contém um inteiro **N** que indica a quantidade de casos de teste. Cada caso de teste inicia com um inteiro **L** ($1 \leq L \leq 100$) que indica a quantidade de linhas que vem a seguir. Cada uma destas **L** linhas contém uma string com até 50 letras maiúsculas ('A' - 'Z').

Saída

Para cada caso de teste imprima o valor de hash que é calculado conforme o exemplo apresentado acima.

Questão 3: Um banco que guarda informações sigilosas de seus usuários, precisa implementar um software para assegurar que os dados de seus clientes estejam seguros contra ataques de hackers. Pensando nisso, você deve escrever um programa orientado a objetos que receba um login e uma senha como dados de entrada, e implemente uma função(função hashing) para “criptografar”a senha(chave) do usuário e associá-la a uma posição de um vetor(endereço).

a. Para isso, você deve converter a senha de no mínimo 6 caracteres e no máximo 10 para um valor hexadecimal.

b. Após a conversão, você deve pegar o primeiro algarismo inteiro do código Hex e usá-lo como endereço para guardar a senha no vetor.

c. E como saída você deve mostrar a senha original, a senha criptografada e a posição no qual a senha se encontra no vetor.