Lista de Exercícios

MC102 – Algoritmos e Programação de Computadores Instituto de Computação Universidade Estadual de Campinas	Modularização, Vetores, Matrizes e Alocação Dinâmica e Recursão 2012 André Santanchè

Questão 1

Escreva uma função em C que receba como parâmetro um vetor e realize uma classificação baseada em merge (*merge sort*). Este tipo de classificação divide recursivamente o vetor em duas partes, até que atinja o tamanho de uma unidade; posteriormente vai realizando fusões (*merges*) progressivos até ordenar todo o vetor original.

Questão 2

Tal como o algoritmo *merge sort*, o algoritmo conhecido como *quick sort* trabalha com recursão e a técnica de dividir para conquistar para realizar sua ordenação. Escreva uma função em C que receba como parâmetro um vetor e o classifique usando a técnica de *quick sort*. Realize uma pesquisa de como funciona conceitualmente o quick sort, sem consultar a respectiva implementação.

Questão 3

Escreva um conjunto de funções em C para dar suporte a um vetor de inteiros que cresce dinamicamente. Considere que existe uma constante `TAXA_CRESCIMENTO` já definida que

`inicializa`	Cria um novo vetor dinâmico e retorna o ponteiro para o mesmo. Esta função não recebe nenhum parâmetro. O tamanho inicial do vetor deve ser igual a `TAXA_CRESCIMENTO` células.
`insere`	Insere um número inteiro no vetor. A função recebe três parâmetros: ponteiro para o vetor dinâmico, inteiro a ser inserido e posição de inserção. Se a posição de inserção for igual a -1 insere no final do vetor. Se a ocupação do vetor tiver que ultrapassar seu limite de tamanho para inserir este novo elemento, ele deve crescer dinamicamente `TAXA_CRESCIMENTO` células.
`exclui`	Exclui um número inteiro do vetor. A função recebe dois parâmetros: ponteiro para o vetor dinâmico e posição de exclusão. Se a posição de exclusão for igual a -1 exclui o último elemento do vetor. Se ao excluir o elemento houver uma sobra de espaço maior que `TAXA_CRESCIMENTO` células, o vetor deve diminuir dinamicamente `TAXA_CRESCIMENTO` unidades.
`finaliza`	Libera a área de memória alocada pelo vetor dinâmico usando a função `free`.
`copia`	Copia todos os elementos de um vetor no final de outro. Recebe como parâmetro o ponteiro para dois vetores dinâmicos A e B. Todos os elementos de A devem ser copiados para o final do vetor B. O vetor B deve crescer dinamicamente se necessário, conforme as regras de inserção.

Questão 4

Considere um polinômio de grau n introduzido na lista de exercícios anterior:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 x^0$$

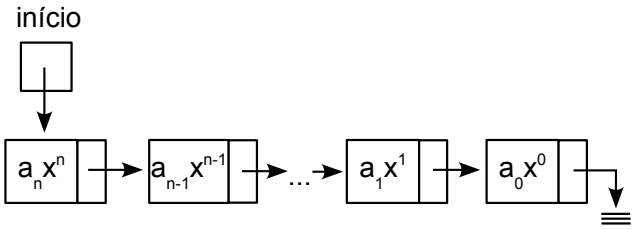
Considere a representação computacional de tal polinômio na qual cada termo $a_i x^i$ é representado por um registro *(struct)*. Conforme apresentado na lista anterior, este polinômio é mantido pelas seguintes funções:

inicializa	Recebe dois parâmetros : a_i e i, e cria um polinômio em memória na forma: $P(x) = a_i x^i$. A função retorna um ponteiro para o vetor contendo o polinômio criado.
termo	Recebe três parâmetros – a_i, i e um ponteiro para a estrutura que representa o polinômio – e adiciona o termo $a_i x^i$ ao polinômio recebido como parâmetro. O polinômio pode ter um terno $a_q x^q$ cujo valor de q seja igual a i, neste caso a função deve unificar ambos em um único termo.
calcula	Recebe um valor de x como parâmetro e retorna o valor de P(x).
finaliza	Libera a área de memória alocada para o polinômio usando a função `free`.
fusao	Recebe como parâmetros dois ponteiros para a estrutura de dois polinômios diferentes e retorna o ponteiro para a estrutura de um novo polinômio que corresponda a fusão dos polinômios de entrada.

Estenda a implementação realizada na lista anterior com duas possíveis variantes:

a) O vetor que representa o polinômio é do tamanho exato do número de termos e cresce dinamicamente cada vez que um novo termo é inserido.

b) O vetor é representado por uma lista encadeada de termos, conforme a figura a seguir. Nesta lista, cada nó consiste em um registro isolado que tem um campo adicional, tipo ponteiro para registro, que aponta para o próximo nó. O último nó aponta para NULL. O acesso ao polinômio é feito por uma variável que aponta para o primeiro nó (variável `início` na figura).



Exercício inspirado em exemplo dos slides de prof. Tomasz Kowaltowski : "Estruturas de Dados e Técnicas de Programação", 2010.

Questão 5

Um programa controla um rato robô dentro de um labirinto. O rato robô funciona diferente de um rato comum, pois além de andar para frente e para trás ele é capaz de andar para as laterais. Ele também possui quatro sensores capazes de identificar paredes nas quatro direções.

Considere que este rato robô está em algum lugar de um labirinto organizado na forma de células quadradas. Cada célula ou possui um espaço completamente vazio, ou possui uma parede ocupando a célula inteira. A meta do rato robô é sair do labirinto.

O rato robô é controlado a partir das seguintes funções:

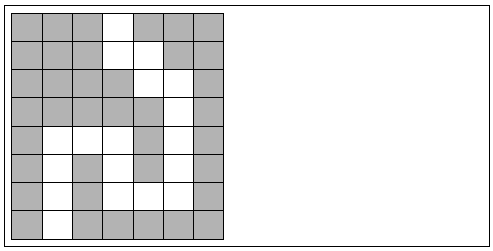
sensor	Ativa um dos quatro sensores de paredes. Recebe como parâmetro uma direção e retorna 0 se a célula vizinha naquela direção estiver vazia e 1 caso haja uma parede.
move	Movimenta o rato robô em uma das quatro direções. Recebe como parâmetro a direção em que se deseja movimentar e se movimenta uma única célula na direção indicada. Se houver parede na direção indicada o rato não se movimenta.
labirinto	Detecta se o rato robô ainda está no labirinto. Não recebe parâmetros e retorna 1 se ele estiver ainda dentro do labirinto e 0 se estiver fora.
desativa	Desativa o rato robô.

Escreva uma função a ser chamada pelo programa de controle que não receba nenhum parâmetro e – acionando as funções da biblioteca acima – controle o rato robô para que ele saia do labirinto. A função

considera que o robô já está dentro do labirinto. É possível que o labirinto não tenha saída. Neste caso, ao detectar isso, a função deve desativar o rato robô.

Considere duas variantes deste problema:

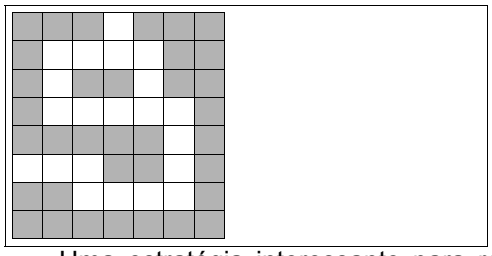
a) Trata-se de um labirinto simples, em formato de túnel, no qual existe um caminho único, sem bifurcações da entrada até a saída. Ele pode ter entre zero e duas saídas, como o exemplo ilustrado a seguir:



Procure desenvolver o programa sem consultar o quadro abaixo de como resolver o problema. Só o consulte se não encontrar uma solução ou depois de ter resolvido o problema.

Para cada posição nova alcançada, o programa deve sempre verificar as outras três posições de destino (excluindo-se a que foi usada para alcançar a posição corrente), a verificação deve seguir o sentido horário.
Cada vez que o programa encontra uma passagem ele segue para a mesma.

b) Trata-se de um labirinto um labirinto mais complexo em que pode haver bifurcações e ciclos, como o exemplo ilustrado a seguir:

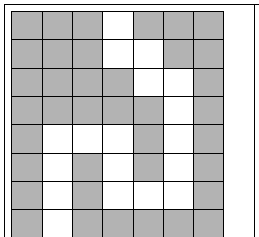


Uma estratégia interessante para resolver este problema envolve uma pilha. Procure desenvolver o programa sem consultar o quadro abaixo de como resolver o problema. Só o consulte se não encontrar uma solução ou depois de ter resolvido o problema.

Para cada posição nova alcançada, o programa deve sempre verificar as outras três posições de destino (excluindo-se a que foi usada para alcançar a posição corrente), a verificação deve seguir o sentido horário.
Cada vez que o programa encontra uma passagem não visitada, ele empilha a posição corrente juntamente com um registro das passagens já visitadas e segue para a nova posição.
Quando o programa chega em um beco sem saída, ou já visitou todas as passagens sem sucesso, ele desempilha a posição de onde veio e retorna. Como o programa já tem o registro das passagens que já visitou, ele deve seguir testando as ainda não visitadas.

Questão 6

Dado um labirinto representado por uma matriz de números inteiros, conforme está ilustrado a seguir:



1	1	1	0	1	1	1
1	1	1	0	0	1	1
1	1	1	1	0	0	1
1	1	1	1	1	0	1
1	0	0	0	1	0	1
1	0	1	0	1	0	1
1	0	1	0	0	0	1
1	0	1	1	1	1	1

Na matriz, as posições que têm o número 1 representam paredes e aquelas com 0 representam

passagens.

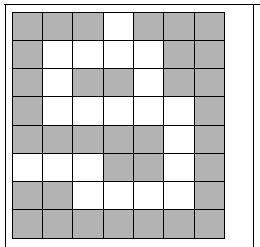
Trata-se de um labirinto simples, em formato de túnel, no qual existe um caminho único, sem bifurcações da entrada até a saída.

Escreva uma função para calcular o caminho dentro do labirinto (não é possível caminhar na diagonal). Ela recebe como parâmetros: a matriz contendo o labirinto, a largura e altura da matriz e as coordenadas X e Y da entrada. A função deve retornar um vetor contendo a sequência de posições no labirinto da entrada até a saída. Cada posição é representada por um par (x, y).

Novamente, considere duas variantes deste problema:

a) Trata-se de um labirinto simples, em formato de túnel, no qual existe um caminho único, sem bifurcações da entrada até a saída. Ele pode ter entre zero e duas saídas, como o exemplo ilustrado no início desta questão.

b) Trata-se de um labirinto um labirinto mais complexo em que pode haver bifurcações e ciclos. Como o exemplo ilustrado a seguir:



1	1	1	0	1	1	1
1	0	0	0	0	1	1
1	0	1	1	0	1	1
1	0	0	0	0	0	1
1	1	1	1	1	0	1
0	0	0	1	1	0	1
1	1	0	0	0	0	1
1	1	1	1	1	1	1