SME0230 - Introdução à Programação de Computadores

Prova final - 25/06/2020

Instruções e observações

Esta prova é **individual**. Suponho que somos todos adultos com um senso ético construído, que permite dissernir o certo do errado. Apenas para reforçar: colar na prova é errado! Provas com questões iguais ou muito similares receberão nota 0.

A entrega da prova será feita através do Run Codes (`https://run.codes`), com código de matrícula da disciplina 4Z51 (mesmo usado para entregar todos os exercícios e trabalhos). Cada uma das duas questões será uma "atividade" diferente do Run Codes.

Alguns cuidados com os códigos que serão entregues são muito importantes. Isso já foi falado e reforçado durante todo o semestre, mas não custa lembrar:

- Comente seu código! Não é para escrever um tratado ou descrever literalmente seu código em português. Os comentários devem ajudar a entender a ideia usada para resolver o exercício e apontar o que trechos de código fazem.
- Deixe seu código claro e organizado. O uso de funções para executar sub-tarefas pode ajudar bastante nesta parte. E não complique seu código à toa (ou para mostrar que sabe alguma coisa). Clareza e boas ideias são muito importantes!
- Seu código precisa resolver os problemas que foram propostos e atender a todos os requisitos pedidos. Não adianta fazer um código que resolve outro problema, caso você não saiba como resolver algum problema proposto. E se alguma coisa for pedida explicitamente, ela precisa ser atendida.
- Não custa lembrar mais uma vez: **NÃO** é permitido usar variáveis globais ou "goto". Códigos com essas coisas receberão nota 0.
- Indentação é **essencial**. Códigos não (ou mal) indentados terão um desconto **considerável** na nota.

Apesar de todas as "pré-broncas", espero que todos se divirtam com a (resolução da) prova. Não entrem em pânico!

Boa prova!

Marina

Questões

1. (3.0) Vamos chamar de *bidigital* qualquer número natural que tenha no máximo dois dígitos distintos.

Por exemplo, 14414 é bidigital, já que só tem os dígitos 1 e 4. O número 555 também, já que só tem o dígito 5. Já o número 109 não é bidigital, já que tem os dígitos 0, 1 e 9.

Vamos chamar de *múltiplo* de um número n todo kn, com $k > 0$ natural.

Sabe-se que todo número natural tem algum múltiplo bidigital. Por exemplo, o primeiro múltiplo bidigital de 14414 é o próprio 14414. O primeiro múltiplo bidigital de 109 é $545 = 109 \times 5$.

Escreva um programa, em linguagem C, que leia um número inteiro $n > 0$ digitado pelo usuário e imprima o primeiro múltiplo bidigital de n. Seu programa deve cuidar do caso em que o valor de n é inválido.

2. (7.0) O jogo "Can't Stop" foi criado por Sid Sackson em 1980. Para jogá-lo são necessários 4 dados, algumas fichas coloridas e um tabuleiro como o da Figura 1 (com 11 colunas, numeradas de 2 a 12, cada uma delas com 3, 5, 7, 9, 11, 13, 11, 9, 7, 5 e 3 casas, respectivamente).

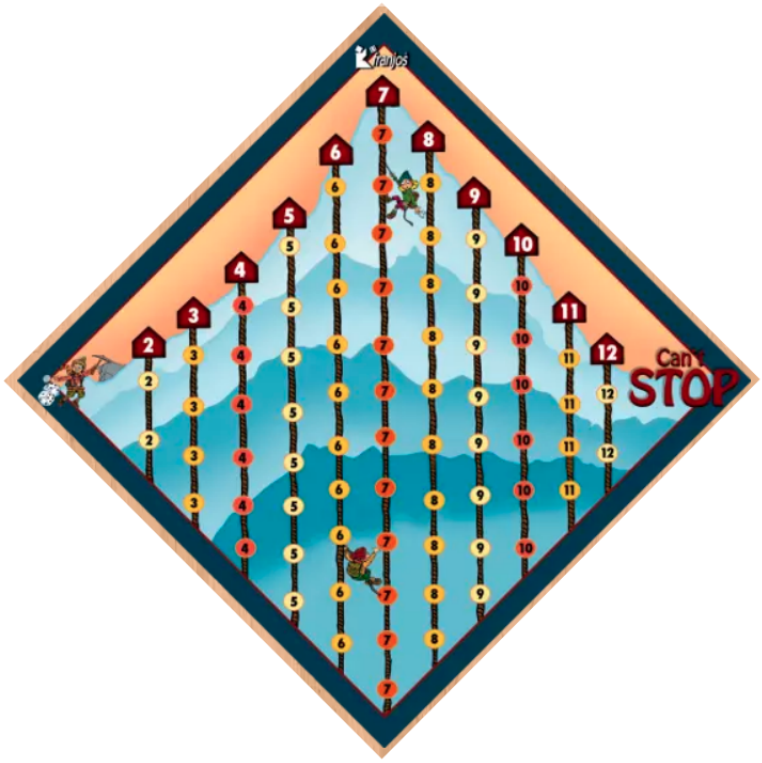

Figura 1: Exemplo de tabuleiro do Can't Stop, retirado de `http://boardgamearena.com`. Último acesso em 22/06/2020.

O jogo permite até 4 jogadores, que alternam suas jogadas entre si, cada um com fichas de uma cor. Na nossa versão haverá apenas 2 jogadores, um com fichas vermelhas e outro com fichas amarelas.

Em cada rodada, cada jogador começa com 3 fichas brancas provisórias e joga os 4 dados. O jogador separa os dados em dois conjuntos com dois dados cada e, para cada conjunto, soma o valor das faces sortedas de seus dados. Chamaremos essas somas de s_1 e s_2. Por exemplo, se os valores sorteados forem 5, 2, 3 e 6, os dados podem ser agrupados como $\{5, 2\}$ ($s_1 = 7$) e $\{3, 6\}$ ($s_2 = 9$); $\{5, 3\}$ ($s_1 = 8$) e $\{2, 6\}$ ($s_2 = 8$); $\{5, 6\}$ ($s_1 = 11$) e $\{2, 3\}$ ($s_2 = 5$).

Se o jogador já possui uma ficha provisória na coluna numerada s_1, essa ficha avança uma casa para cima. Se o jogador não possui ficha branca nesta coluna, mas possui alguma em sua mão, deve colocá-la nesta coluna.

Se a coluna numerada s_1 já tem uma ficha permanente da cor de seu jogador (vermelha para o Jogador 1 e amarela para o Jogador 2), a ficha branca é colocada na linha imediatamente acima da ficha permanente do jogador. Se não há fichas permanentes do jogador na coluna escolhida, a ficha branca é colocada na primeira linha (de baixo para cima) da coluna.

O mesmo procedimento deve ser feito com a coluna numerada s_2 (quando possível), mesmo que esse valor seja igual a s_1.

Quando o jogador atual termina sua jogada, ele tem de escolher se quer fazer outra jogada (mas sem receber novas fichas provisórias) ou se quer parar.

Se o jogador decide parar, todas as suas fichas provisórias que estão no tabuleiro são trocadas por fichas permanentes. Se há duas fichas permanentes em uma coluna, apenas a que está em posição mais alta é mantida.

Neste momento, se uma ficha permanente atinge o topo de alguma coluna, aquela coluna passa a "pertencer" ao jogador e a coluna é "fechada", o que faz com que a ficha do outro jogador que esteja nesta coluna seja eliminada e ninguém mais possa colocar fichas nesta coluna.

Depois de um jogador jogar os dados, se não há jogada possível para ele (ou seja, não consegue jogar nem na coluna s_1, nem na s_2, seja porque as colunas estão "fechadas", seja porque o jogador não possui mais fichas brancas na mão e suas fichas brancas no tabuleiro não estão nem na coluna s_1, nem na s_2), todas as fichas brancas do jogador são eliminadas do tabuleiro e a vez passa para o outro jogador.

O jogo termina quando 3 colunas forem "fechadas" por um mesmo jogador. Vence o jogo quem fechar essas 3 colunas.

Observação: é possível ter mais de uma ficha em uma mesma casa!

Esse jogo esta disponível, por exemplo, em `http://boardgamearena.com/gamepanel?game=cantstop`. Lá é possível jogar, ver as regras do jogo e ainda há tutoriais disponíveis.

Implemente, em linguagem C, o jogo Can't Stop com 2 jogadores.

Você pode supor que o usuário sabe as regras do jogo. Mas deve ficar claro o que ele deve fazer para poder jogar (como escolher os valores sorteados nos dados ou as somas s_1 e s_2, como escolher se vai continuar jogando, etc). Casos em que o usuário pode digitar valores errados devem ser previstos.

Note que, a partir do momento que o usuário define os valores de s_1 e s_2 que vai querer usar, pode ser que ele não tem mais nenhuma escolha no posicionamento das fichas. Neste caso, esse posicionamento deve ser feito automaticamente.

Dica: você pode usar ideias da "corrida de letras" feita em aula para implementar este jogo. Neste caso, em vez de armazenar a posição em que cada letra está na pista, você pode armazenar a posicao em que tem fichas de cada cor para cada coluna do tabuleiro.