Prova 1 (2020/02)

Valor: 15 pontos

Ao concordar em fazer esta prova, eu juro que seguirei o **código de honra**:

1. Não ajudarei e nem pedirei ajuda a ninguém durante o período de submissão do exame;
2. Não divulgarei qualquer informação sobre as minhas soluções durante o período de submissão do exame.

É normal não entender completamente uma questão. Se você tiver dúvidas sobre qualquer questão, envie uma mensagem ***PRIVADA*** no Teams ou um e-mail para olmo@dcc.ufmg.br com o assunto **[PROVA1]** e explique a sua dúvida para mim. Tentarei responder o mais rápido possível, mas a resposta pode demorar se a mensagem for enviada fora do horário comercial.

Sumário

Nesta prova, você vai implementar funções que compõem um jogo de mesa simples. Nesse jogo de dois jogadores, palitos são dispostos em três pilhas, que podem ter tamanhos diferentes. As pilhas são identificadas por um número inteiro, ou pilha **1**, pilha **2** e pilha **3**. O jogo ocorre em turnos alternados, e em cada turno o jogador escolhe uma pilha e uma quantidade de palitos para remover dessa pilha. O jogador que deixar a mesa sem palitos para o seguinte perde o jogo, ou seja, normalmente o jogo termina com o jogador perdedor removendo o último palito da mesa. Algumas regras sobre o jogo:

1) O jogador só pode remover palitos de uma única pilha;
2) O jogador pode remover quantos palitos quiser (de uma única pilha);
3) O jogador é obrigado a escolher uma pilha que tenha pelo menos 1 palito;
4) Se o jogador escolher uma pilha ou um número de palitos inválidos, ele terá que fazer uma nova escolha;
5) Se o jogador remover todos os palitos de uma pilha e essa for a última pilha com palitos disponível, ele perde o jogo.

Importante: você pode criar quantas funções extras desejar.

Exercícios

1) Implemente uma função de nome **`imprimePilhas`** que não retorna nada (**void**) e que recebe três inteiros como parâmetros, **p1**, **p2** e **p3**, que correspondem à quantidade de palitos na **pilha 1**, **pilha 2** e **pilha 3**, respectivamente. A sua função deve imprimir, em ordem e para cada pilha, a quantidade de palitos na pilha e, em seguida, cada um dos palitos na pilha usando o caractere `|` (barra vertical, ou *pipe*, é o caractere ASCII de código 124). Por exemplo, se **p1=6**, **p2=1** e **p3=4**, a sua função deve imprimir:

```
6 ||||||
1 |
4 ||||
```

Importante: há um espaço entre as quantidades e o primeiro caractere `|`. Além disso, inicie e termine a impressão com uma quebra de linha (`\n`). Protótipo:

```
void imprimePilhas (int p1, int p2, int p3);
```

Valor: **2 pontos**.

2) Implemente uma função de nome **`imprimePilhas2`** que não retorna nada (**void**) e que recebe três inteiros como parâmetros, **p1**, **p2** e **p3**, que correspondem à quantidade de palitos na **pilha 1**, **pilha 2** e **pilha 3**, respectivamente. Diferente da função anterior, aqui você deve imprimir a representação vertical dos palitos na mesa. Além disso, na base de cada pilha, você deve imprimir a quantidade de palitos que a pilha possui. Como a representação é vertical, ao invés do caractere `|`, você deve usar o caractere `_` (*underline*, ou *underscore*, é o caractere ASCII de código 95). Por exemplo, se **p1=6**, **p2=1** e **p3=4**, a sua função deve imprimir:

```
_
_
_   _
_   _
_   _
_ _ _
6 1 4
```

Importante: há um espaço entre as pilhas. Inicie e termine a impressão com uma quebra de linha (`\n`). Não se preocupe em manter as pilhas de palitos alinhadas com os números na sua base caso o número tenha dois dígitos ou mais. Protótipo:

```
void imprimePilhas2 (int p1, int p2, int p3);
```

Valor: **3 pontos**.

3) Implemente uma função de nome **`leJogada`** que não retorna nada (**void**) e recebe dois endereços de memória para inteiros, `end_p` e `end_q`. A sua função deve pedir dois inteiros do usuário e armazenar o primeiro em `end_p` e o segundo em `end_q`. No jogo, os inteiros armazenados nesses endereços correspondem ao número da pilha (**1**, **2** ou **3**) e a quantidade de palitos que o usuário gostaria de remover dessa pilha. Assim, antes de ler o inteiro a ser armazenado em `end_p`, imprima a frase `"Escolha uma pilha (1, 2 ou 3):\n"`. Antes de ler o inteiro a ser armazenado em `end_q`, imprima a frase `"Quantos palitos gostaria de remover?\n"`. Não se preocupe em testar se o número da pilha e a quantidade de palitos são valores válidos. Protótipo:

```
void leJogada (int *end_p, int *end_q);
```

Valor: **2 pontos**.

4) Implemente uma função de nome **`diminuiPilha`** que retorna um inteiro (**int**) e recebe um endereço de memória para inteiro `end_p` e um inteiro `q`. Nesta função, você deve tentar subtrair `q` do inteiro armazenado em `end_p`. No jogo, o inteiro armazenado em `end_p` corresponde ao número de palitos de uma pilha e `q` corresponde à quantidade de palitos que deve ser removido dessa pilha. Se `q` for uma quantidade válida, você deve subtrair `q` do inteiro armazenado em `end_p` e depois retornar **1**. Caso contrário, simplesmente retorne **0**. A quantidade `q` não é válida se ela for menor ou igual a **0** ou maior que o inteiro armazenado em `end_p`. Protótipo:

```
int diminuiPilha(int *end_p, int q);
```

Valor: **2 pontos**.

5) Implemente uma função de nome **`confereJogo`** que retorna um inteiro (**int**) e que recebe três inteiros como parâmetros, **p1**, **p2** e **p3**, que correspondem à quantidade de palitos na **pilha 1**, **pilha 2** e **pilha 3**, respectivamente. A sua função deve verificar se o jogo acabou ou não. Caso haja algum palito na mesa, a sua função retorna 1. Caso contrário, ela retorna 0. Protótipo:

```
int confereJogo(int p1, int p2, int p3);
```

Valor: **1 ponto**.

6) Implemente uma função de nome **`jogo`** que retorna um inteiro (**int**) e que recebe três inteiros como parâmetros, **p1**, **p2** e **p3**, que correspondem à quantidade de palitos na **pilha 1**, **pilha 2** e **pilha 3**, respectivamente. A sua função deverá executar o jogo entre os dois jogadores e retornar o identificador do jogador vencedor. O identificador do primeiro jogador a jogar é **1** e o do segundo é **2**. Cada jogada será executada de acordo com o seguinte processo. No início de cada jogada, a função deve imprimir a quantidade de palitos em cada pilha, usando o seguinte formato: `"p1 p2 p3\n"` (exemplo: `"4 1 6\n"`). Sugestão: se usar a função `imprimePilha2`, essa impressão é feita automaticamente. Depois, a função deve imprimir o identificador do jogador que fará a jogada e pedir uma jogada a ele. Em uma jogada, o jogador deve informar primeiro o número da pilha e depois a quantidade de palitos a ser removida. Sugestão: use a função `leJogada`. Depois, você deve reduzir, caso seja possível, a quantidade escolhida de palitos da pilha escolhida por ele. Sugestão: use a função `diminuiPilha`. Caso a jogada tenha sido viável, repita todo o processo para o jogador seguinte. Caso contrário, imprima a mensagem erro `"\nErro! Jogue novamente!\n"` e repita todo o processo para o mesmo jogador. A sua função deve retornar o identificador do jogador que venceu o jogo, ou seja, o jogador que não pegou o(s) último(s) palito(s) disponível(is) na mesa. Sugestão: use a função `confereJogo` para verificar se o jogo acabou ou não.

```
int jogo(int p1, int p2, int p3);
```

Valor: **4 pontos**.

7) Implemente uma função de nome **`principal`** que retorna um inteiro (**int**) e que não tem parâmetros. Esta função simula a função **"main"** do seu programa. Nesta função, você deve ler três inteiros do usuário, que correspondem à quantidade inicial de palitos na **pilha 1**, **pilha 2** e **pilha 3**, respectivamente. Enquanto qualquer dos valores lidos for menor ou igual **0**, você deve repetir esse processo, ou seja, pedir novos três inteiros para o usuário. Depois, você deve executar o jogo a partir da função `jogo`. A sua função deve retornar identificador correspondente ao vencedor do jogo. Protótipo:

```
int principal();
```

Valor: **1 ponto**.

Execução no VPL

Para conseguir compilar o seu programa, você deve implementar uma versão sintaticamente correta de todas as funções pedidas. Sugiro fortemente que faça isso antes de pensar nas soluções. Para o exercício **3**, por exemplo, você pode implementar a seguinte função:

```
int confereJogo(int p1, int p2, int p3) { return 0; }
```

Além disso, sugiro que avalie (clicando no botão “avaliar” do VPL) o programa dessa forma para poder visualizar todos os testes disponíveis para esta prova. Você pode avaliar o seu programa quantas vezes quiser.

Para testar um exercício, inicie a execução no VPL e digite o número da função que gostaria de testar (**1**, **2**, **3**, **4, 5, 6** ou **7**). Para as funções **1, 2, 5** e **6,** digite também a quantidade de palitos em cada pilha. Para a função **4**, digite dois valores inteiros, que são os parâmetros da função. Para a função **3**, você só precisa digitar os valores pedidos na própria função. Por exemplo, se quiser testar a função `imprimePilhas` (exercício **1**) com quantidades equivalentes a **p1=6**, **p2=1** e **p3=4**, digite:

```
1
6 1 4
```