MC202 — ESTRUTURAS DE DADOS

Laboratório 09 — Blackjack

Blackjack, também conhecido como 21, é um jogo de cartas, jogado em mesa com vários jogadores. Cada um aposta individualmente contra o dealer – jogador que dá as cartas. O jogo utiliza um baralho normal de quatro naipes e cada naipe contém treze cartas (Ás, 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei). O objetivo consiste em chegar o mais próximo possível de 21 pontos, sem ultrapassar esse valor. O Ás vale 1 ponto ou 11 pontos, o jogador sempre escolherá 11 a não ser que ele for estourar os 21 pontos. As cartas numeradas valem seu respectivo valor: uma carta 2 vale 2 pontos e assim por diante. As cartas com figuras (Valete, Dama e Rei), valem 10 pontos cada uma.

No início do jogo, o dealer dá uma carta para cada jogador, da esquerda do dealer para a direita, incluindo ele mesmo. Em seguida, o dealer dá outra carta a cada jogador, no mesmo sentido, sempre incluindo ele mesmo.

Então, os jogadores (incluindo o dealer) que tiverem menos de 21 pontos tomam turnos para fazer suas decisões. Em cada rodada, o dealer oferece mais uma carta a cada jogador e a ele mesmo, contanto que não possuam mais de 20 pontos. Isto é, o jogador pode aceitar ou recusar cartas enquanto ele tiver menos do que 21 pontos.

Quando um jogador aceita uma carta, dizemos que houve um hit. Se ainda possuir menos de 20 pontos, o jogador ou o dealer que considerar que possui todas as cartas de que precisa, ele recusa a carta oferecida e permanece com sua mão sem aceitar cartas adicionais; nesse caso, dizemos que ele está em $stand$.

Ou seja, nesta versão, se a mão de um jogador ou do dealer somar 21 ou mais pontos, então não será oferecida carta para ele, mesmo que ele não esteja em $stand$.

O dealer dispõe de algumas cartas para trapacear enquanto os jogadores não estiverem atentos a pilha de cartas. Assim, a cada movimento, o dealer pode inserir uma dessas cartas no topo da pilha.

Uma vez que todos os jogadores e o dealer estiverem servidos, os pontos são contabilizados. Se a mão de um jogador exceder 21 pontos, ele perde instantaneamente, independentemente da mão do dealer. Do contrário, se a mão do dealer exceder 21 pontos, o jogador ganha. Se nenhum dos dois exceder 21 pontos, ganha quem chegar mais perto de 21 pontos.

Tarefa

A sua tarefa consiste em desenvolver um programa que, dada uma sequência de cartas que compõem o baralho e os movimentos realizados por cada jogador e pelo dealer, determine a pontuação de cada um deles.

A sequência de cartas que compõem o baralho deve ser mantida em uma estrutura de dados do tipo LIFO, onde a última carta a ser inserida será a primeira a ser retirada. Já os jogadores devem ser mantidos em uma estrutura de dados do tipo FIFO, onde o primeiro jogador a ser inserido, será o primeiro a ser retirado.

Entrada

A primeira linha contém dois números inteiros $m \geq 26$ e $n \geq 1$, que indicam respectivamente o número de cartas do baralho e número de jogadores. A próxima linha contém m strings, separadas por espaço, que representam as cartas a serem empilhadas. As cartas são distinguidas apenas por seus valores, representados, respectivamente, por "A", "2", "3", …, "10", "V", "D" e "R". As próximas linhas representam os movimentos realizados durante o jogo, sendo "S" e "H", jogador com menos de 21 pontos fica em stand ou aceita uma carta respectivamente, "c", dealer colocar a carta c no topo da pilha, e "#", fim de execução.

Saída

O seu programa deve produzir n+1 linhas contendo a soma dos pontos obtidos por cada jogador, em ordem, e do dealer.

Exemplo

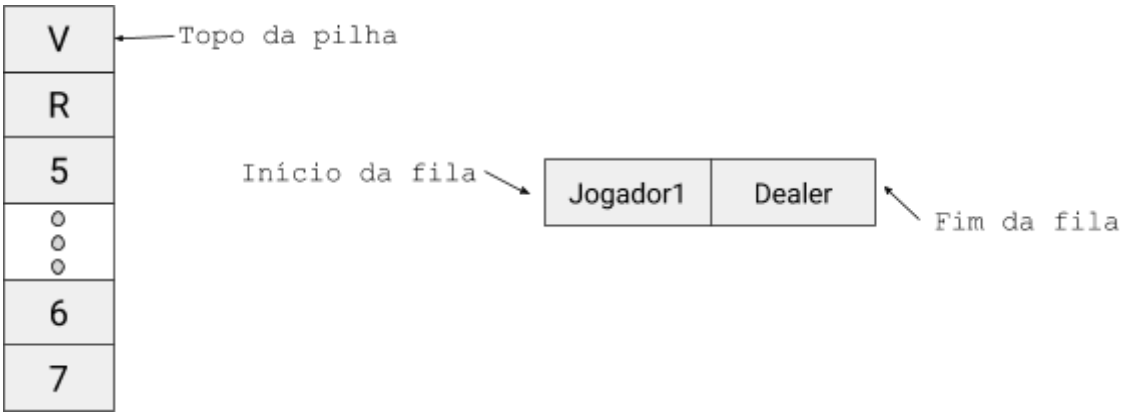
Entrada 1

```
26 1
7 6 5 2 A 6 4 R 8 D V 4 10 3 10 7 A D 9 9 3 2 8 5 R V
H
H
S
#
```

Saída 1

```
17
21
```

Representação



Execução

```
Jogador1 : recebe a carta V
Dealer   : recebe a carta R
Jogador1 : recebe a carta 5
Dealer   : recebe a carta 8
Jogador1 : hit -> compra a carta 2
Dealer   : hit -> compra a carta 8
Jogador1 : stand
#
Jogador1 : fez 17 pontos
Dealer   : fez 21 pontos
```

Entrada 2

```
26 3
7 6 5 2 A 6 4 R 8 D V 4 10 3 10 7 A D 9 9 3 2 8 5 R V
H
H
H
3
H
S
S
#
```

Saída 2

```
22
14
21
20
```

Execução

```
Jogador1 : recebe a carta V
Jogador2 : recebe a carta R
Jogador3 : recebe a carta 5
Dealer   : recebe a carta 8
Jogador1 : recebe a carta 2
Jogador2 : recebe a carta 3
Jogador3 : recebe a carta 9
Dealer   : recebe a carta 9
Jogador1 : hit -> compra a carta D
Jogador2 : hit -> compra a carta A
Jogador3 : hit -> compra a carta 7
Dealer   : colocar na pilha a carta 3
Dealer   : hit -> compra a carta 3
Jogador2 : stand
Dealer   : stand
#
Jogador1 : fez 22 pontos
Jogador2 : fez 14 pontos
Jogador3 : fez 21 pontos
Dealer   : fez 20 pontos
```

Dicas

Você pode ler as cartas e os movimentos usando `scanf("%s", ...)` e quando for um número utilizar a função `atoi()` para converter a string lida em um inteiro.

Lembre-se de que a pontuação de um jogador depende do conjunto de cartas e de áses que ele possui. Se preferir, você **pode** (não é obrigatório) manter um vetor ou uma lista encadeada das cartas de cada jogador.

Critérios específicos

- Para as turmas E e F, este laboratório tem peso 3.
- Para as turmas G e H, este laboratório tem peso 2.
- Deverão ser submetidos os seguintes arquivos: **lab09.c**, **fila.***, **pilha.*** e **jogador.***.
- Tempo máximo de execução: 1 segundo.

Testando

Para compilar com o Makefile fornecido e verificar se a solução está correta basta seguir o exemplo abaixo.

```
make
./lab09 < arq01.in > arq01.out
diff arq01.out arq01.res
```

onde `arq01.in` é a entrada (casos de testes disponíveis no SuSy) e `arq01.out` é a saída do seu programa. O Makefile também contém uma regra para testar todos os testes de uma vez; nesse caso, basta digitar:

```
make testar_tudo
```

Observações gerais

No SuSy, haverá 3 tipos de tarefas com siglas diferentes para cada laboratório de programação. Todas possuirão os mesmos casos de teste. As siglas são:

1. **SANDBOX:** Esta tarefa serve para testar o programa no SuSy antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém arquivos submetidos aqui **não serão corrigidos**.

2. **ENTREGA:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa: submeta aqui quando não for mais fazer alterações no seu programa.

3. **FORAPRAZO:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final após o prazo estabelecido para o laboratório, mas com nota reduzida (conforme a ementa). O envio nesta tarefa irá substituir a nota obtida na tarefa ENTREGA apenas se o aluno tiver realizado as correções sugeridas no feedback ou caso não tenha enviado anteriormente em ENTREGA.

Observações sobre SuSy:

- Versão do GCC: C-ANSI 4.8.2 20140120 (Red Hat 4.8.2-15).

- Flags de compilação:
  `-ansi -Wall -pedantic-errors -Werror -g -lm`

- Utilize comentários do tipo `/* comentário */`;
  comentários do tipo `//` serão tratados como erros pelo SuSy.

Além das observações acima, esse laboratório será avaliado pelos critérios gerais:

- Indentação de código e outras boas práticas, tais como:
    - uso de comentários (apenas quando forem relevantes);
    - código simples e fácil de entender;
    - sem duplicidade (partes que fazem a mesma coisa).
- Organização do código:
    - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.
- Corretude do programa:
    - programa correto e implementado conforme solicitado no enunciado;
    - inicialização de variáveis sempre que for necessário;
    - dentre outros critérios.