Universidade Federal do Espírito Santo – Centro Tecnológico
Departamento de Informática
Prof. Thiago Oliveira dos Santos

UFES

Instruções gerais: A prova tem o objetivo de avaliar os conceitos ensinados e trabalhados até o momento. Ela deve ser feita individualmente e sem ajuda de terceiros. Casos de cola, terceirização, ajuda para fazer a avaliação ou qualquer outro que beneficie o aluno de forma indevida serão tratados com a penalidade máxima disponível na instituição (i.e., abertura de processo administrativo podendo levar a expulsão). A penalidade será aplicada com igual teor a todos os envolvidos e sem distinção (inclusive a quem fornecer). Siga as regras e faça sua própria prova usando os conceitos ensinados em sala. Boa prova!

Valor: 5 pontos

Correção: a questão está dividida em cinco partes de teste, *a*, *b*, *c*, *d* e *e*, sendo que a complexidade dos casos aumenta com as letras. Os casos das letras posteriores contemplam os casos das anteriores e mais alguns. Por exemplo, os casos de *b* contemplam os de *a*, os de *c* contemplam os de *a* e *b* e, assim por diante. Portanto, se resolver a parte *e*, resolveu toda a questão. Todas as partes possuem casos de teste ocultos não fornecidos com a questão. O Anexo I descreve os tipos de casos com mais detalhes. O aluno ganha os pontos totais se obtiver o balão na parte *e* (que inclui todos os casos) e ganha 20%, 40%, 60% e 80% se passar somente até a parte *a*, *b*, *c* ou *d*, respectivamente. A questão a ser corrigida será a última enviada, dando prioridade a que estiver correta. Portanto, sempre envie a sua solução para as duas partes. Não implementar as funções (subprogramas) obrigatórias pedidas abaixo levará a descontos de até 20% do valor da questão.

(**BOCA: P1_2021_Q1**) Problema: Um homem queria criar senhas numéricas para suas contas em sites de compras. Ele sabia que não conseguiria guardar as senhas de cabeça e temia salvá-las em um arquivo qualquer em seu computador por medo de ser hackeado. Então ele resolveu implementar um código gerador de senha aleatório que criaria arquivos absurdos aos olhos de quem não soubesse o segredo para decodificá-lo. Seu objetivo nessa questão é, a partir de um arquivo de texto criado pelo gerador de senha, implementar um programa que decodifique a senha passo a passo.

Exemplo de um arquivo de texto qualquer criado pelo gerador de senha (a coluna à esquerda apenas indica o número da linha no arquivo de texto):

1	d = 1.
2	? d.
3	#
4	d = 11 2 M.
5	e = 1 2 A d D 9 A.
6	? d e A.
7	d = 5.

```
8   e = 4 3 A.
9   #
10  ? 1 1 A.
11  a = 3 4 M 2 D.
12  ? a a A 2 M.
13  #
14  a = 5 6 M.
15  ? a a M 1 A.
16  b = 1 a A 4 D.
17  a = b 1 A.
18  b = b a D.
19  b = b 5 A 7 M 3 D.
20  ? b.
21  ? a a M c D.
22  ? 1 1 A 2 D.
23  #
24  a = 5.
25  #
26  a = 1.
27  ? 1 1 A a A.
28  #
29  c = 2 2 M 3 A.
```

Saída esperada a partir do exemplo de entrada dado acima:

```
COMECOU
31
TERMINOU

COMECOU
901
11
IGNORAR RESTANTE DESTE TRECHO
TERMINOU

COMECOU
3
TERMINOU

SENHA: 921723
```

A decodificação segue as regras:

1) Apenas certos trechos da entrada são relevantes para a decodificação. Os inícios dos trechos relevantes são marcados por uma linha contendo apenas o caractere “#” e o fim

dos trechos relevantes também. Cada "#" representa o início **ou** o fim de um trecho, **nunca ambos**. Fora dos trechos relevantes, todo o resto deve ser ignorado. Por exemplo, no arquivo de entrada acima, as linhas de 4 a 8, 14 a 22 e 26 a 27 são trechos relevantes, todo o resto deve ser ignorado. Ao início de um trecho relevante, deve ser impressa a mensagem "COMECOU" e ao final do trecho, a mensagem "TERMINOU" seguida de uma linha em branco.

2) A entrada sempre possui uma formatação padrão, definida pelas seguintes regras:

2.1) Todas as linhas sempre terminam com um ponto final (com exceção das linhas que contém apenas "#") e todos os caracteres são separados por um espaço, com exceção do ponto final que vem logo depois do último caractere da linha.

2.2) Apresenta, no máximo, 5 variáveis, representadas pelas cinco primeiras letras do alfabeto minúsculas ("a", "b", "c", "d", "e"). Nunca aparecerão outras variáveis além dessas.

2.3) Contém expressões sempre escritas com notação pós-fixada (i.e. operando da esquerda, operando da direita e operador).

2.4) Um operador é sempre adição (indicado por "A"), multiplicação (indicado por "M"), ou divisão inteira (indicado por "D"). Nunca são realizadas operações de subtração.

2.5) Cada linha segue um dos seguintes padrões, onde "var" é variável, "exp" é expressão e "num" é um número inteiro positivo (sem sinal). Por exemplo:

- var = exp.
- var = var.
- var = num.
- ? exp.
- ? var.
- ? num.

→"var =" significa que a variável recebe o valor da expressão ou variável que vem a seguir.

→ "? " manda imprimir o valor da expressão ou da variável que vem a seguir.

→ Toda "exp" (expressão) segue um dos seguintes padrões, onde "var" é variável, "opr" é um operador ("A", "M" ou "D"), "num" é um número inteiro positivo (sem sinal) e "exp" é uma expressão anterior, usada como operando da esquerda na exp atual. Por exemplo:

- var var opr
- num var opr
- var num opr
- num num opr
- exp var opr
- exp num opr

2.6) Como consequência das duas regras anteriores (i.e., "num" ser sempre inteiro positivo e não haver operação de subtração), **nenhuma expressão gerará resultado menor que zero.** Além disso, **nenhuma variável será negativa.**

3) A precedência dos operadores deve ser desconsiderada, usando a ordem da esquerda para direita como precedência. Portanto, o resultado da primeira operação (da esquerda para a direita) será sempre utilizado como operando da esquerda da próxima operação.

→a b A	representa	*(a + b)*
→a b A c D	representa	*((a + b) / c)*
→a b M c D d A e M	representa	*((((a * b) / c) + d) * e)*

4) A cada novo trecho, todas as variáveis são renovadas, e cada variável, se usada, deve obrigatoriamente ser inicializada primeiro dentro daquele trecho. Se uma variável for invocada em um momento qualquer sem ter sido inicializada naquele trecho, o programa deve ignorar todo o restante daquele que seria um trecho relevante e imprimir a mensagem "IGNORAR RESTANTE DESTE TRECHO". Essa situação pode ser vista no exemplo dado acima. A linha 21 tenta utilizar a variável c sem estar inicializada. Isso gera a impressão da mensagem referida anteriormente, fazendo também ignorar o restante do trecho (isto é, a linha 22).

5) A senha gerada pelo programa ao final é o resultado da multiplicação de todos os valores numéricos impressos ao longo do programa. Ela deve ser impressa ao final: "SENHA: @", sendo @ o valor da senha. (OBS: não haverá valores de senha válidos grandes o suficiente para estourar o limite máximo do tipo **int** do C).

OBS: Seu código deve ser modularizado. Por isso, obrigatoriamente, utilize:

→Uma função de cabeçalho *int CalculaPosFixado(int v1, int v2, char opr)* para resolver expressões do tipo simples, isto é dois operandos e um operador.

→ Uma função de cabeçalho *int ReduzExpressao(int a, int b, int c, int d, int e)* que resolve a expressão inteira da linha (tudo que vem após um "? " ou "var = "). Assuma que está função deve ser chamada no início de uma expressão e finaliza quando consumir o caractere ".".

→Opcionalmente, outras funções podem ser implementadas.

Universidade Federal do Espírito Santo – Centro Tecnológico
Departamento de Informática
Prof. Thiago Oliveira dos Santos

UFES

ANEXO I

A correção da questão será feita de forma incremental, onde cada caso de teste apresentado cobra certos conceitos além do caso anterior, permitindo que a nota também seja incremental.

a) Caso de teste 1 (Apenas um bloco; sem variáveis, só impressões)

Entrada:

```
#
? 100 19 M 37 D.
? 1 2 A 3 M 1 A.
? 7 90 D 3 A.
? 2.
? 1 1 A 1 A 1 A 1 A 1 A 1 A 2 M.
#
```

Saída esperada:

```
COMECOU
51
10
3
2
14
TERMINOU

SENHA: 42840
```

b) Caso de teste 2 (Apenas um bloco; com uso de variáveis, mas todas inicializadas)

Entrada:

```
#
d = 2.
e = 7 8 A 3 M d D.
b = 190 76 A.
c = 1 1 A.
d = c.
? b e D 3 M.
a = 7 7 M 2 A.
b = 17 18 D.
b = b 4 A 3 A.
? b 7 A.
e = d d A d M d A d M 5 A.
? e 1 A.
```

```
#
```

Saída esperada:

```
COMECOU
36
14
26
TERMINOU

SENHA: 13104
```

c) Caso de teste 3 (Apenas um bloco; com uso de variáveis não-inicializadas)

Entrada:

```
#
d = 3.
e = 7 8 M d D 1 D.
b = 22 88 A d A.
? b.
c = d.
? b b D c A 11 M.
c = 1 2 A a A.
a = 2 2 A.
c = 3 5 M d D.
? e.
? c.
e = 99 b D.
? e.
#
```

Saída esperada:

```
COMECOU
113
44
IGNORAR RESTANTE DESTE TRECHO
TERMINOU

SENHA: 4972
```

d) Caso de teste 4 (Múltiplos blocos, com uso de variáveis não-inicializadas, porém um bloco posterior não utiliza uma variável não inicializada que já foi inicializada em outro bloco anterior)

Entrada:

```
d = 1.
? d.
#
d = 11 2 M.
e = 1 2 A d D 9 A.
? d e A.
d = 5.
e = 4 3 A.
#
? 1 1 A.
a = 3 4 M 2 D.
? a a A 2 M.
#
a = 5 6 M.
? a a M 1 A.
b = 1 a A 4 D.
a = b 1 A.
b = b a D.
b = b 5 A 7 M 3 D.
? b.
? a a M c D.
? 1 1 A 2 D.
#
a = 5.
#
a = 1.
? 1 1 A a A.
#
c = 2 2 M 3 A.
#
c = 1.
#
a = c 1 A 4 M.
```

Saída esperada:

```
COMECOU
31
TERMINOU

COMECOU
901
11
IGNORAR RESTANTE DESTE TRECHO
TERMINOU

COMECOU
```

```
3
TERMINOU

COMECOU
TERMINOU

SENHA: 921723
```

e) Caso de teste 5: (Múltiplos blocos, com uso de variáveis não-inicializadas, um bloco posterior utiliza variáveis não inicializadas que já foram inicializadas em outro bloco anterior).

Entrada:

```
d = 1 1 A.
? d d A.
#
d = 11 2 M 1 A.
e = 24 20 A d D 1 M.
d = d 3 A 2 A.
? d e A.
? d.
d = d d D.
d = d e A.
? d.
#
? 1 1 A 1 A.
c = 3 a A 2 D.
? c c A 2 M.
#
c = 3 4 M 2 D.
? c.
#
d = 1 1 A.
#
b = c 2 M.
#
a = 7 6 M 2 M 3 M.
? a.
#
a = 2 2 M.
? a.
c = 12 25 A.
? c a D.
? b b M.
b = 1 c A 4 D.
```

```
c = b 1 A.
a = b c D 2 A.
? b.
b = c c D 111 A.
? b.
? 1 1 A 2 D 1 M.
#
a = 5.
```

Saída esperada:

```
COMECOU
29
28
2
TERMINOU

COMECOU
6
TERMINOU

COMECOU
IGNORAR RESTANTE DESTE TRECHO
TERMINOU

COMECOU
4
9
IGNORAR RESTANTE DESTE TRECHO
TERMINOU

SENHA: 350784
```