Universidade de São Paulo

Instituto de Ciências Matemáticas e de Computação

Departamento de Sistemas de Computação

Disciplina: SSC - 0502

Professor: Claudio Fabiano Motta Toledo

Prova 1

Instruções Gerais

- Leiam atentamente os exercícios antes de começarem a implementar. Dúvidas serão respondidas apenas nos 20 primeiros minutos.

- Leiam atentamente as saídas de erro do run.codes para poder resolvê-los corretamente e seu programa funcionar corretamente.

- Apenas programas com comentários claros terão seu código considerado na correção em caso de erros na saída. Se seu código não conseguir passar em todos os casos testes e não estiver comentado, apenas a nota do run.codes será considerada.

- Os códigos serão verificados se as condições pedidas nos exercícios foram garantidas (por exemplo, o uso de funções). Caso alguma condição não tenha sido cumprida, haverá desconto na nota.

- Será verificado plágio dos códigos, portanto, não tentem colar durante a prova. Qualquer detecção de plágio resultará em nota 0 para ambos os projetos.

Exercício 1 (Valor: 3.0 pontos):

Você deverá fazer um programa para identificar, através das coordenadas dos 4 pontos de um quadrilátero, se ele é um quadrado, ou seja, se todas as arestas possuem o mesmo tamanho.

Você receberá 8 valores inteiros como entrada: as coordenadas x e y de cada ponto.

Os dois primeiros valores serão as coordenadas x e y, respectivamente, do vértice superior esquerdo. Os dois próximos valores, as coordenadas do vértice superior direito. Em seguida o x e o y do vértice inferior direito e, por fim, as coordenadas do vértice inferior esquerdo.

Você deverá imprimir "Quadrado'\n'" caso as arestas tenham mesmo tamanho. Caso contrário, imprima "Quadrilatero'\n'" (sem acento!). Não esqueça-se do caractere '\n' ao final!

Obs: Para simplificar o exercício, estamos chamando de **Quadrado**, genericamente, os **losangos e quadrados** em si. Porém, existem condições especiais para cada um desses quadriláteros! Ou seja, vocês não precisam se preocupar com essa diferença neste exercício.

Exemplo:

Entrada:

0 1
1 1
1 0
0 0

Saída:

Quadrado'\n'

Entrada:

0 1
2 2
1 0
0 0

Saída:

Quadrilatero'\n'

Exercício 2 (Valor: 3.0 pontos):

Você está gerenciando a alocação de blocos de arquivos no HD do seu computador. Você sabe que o HD é dividido em blocos de determinado tamanho.

Se um arquivo for maior que o espaço restante no bloco, ele teria de ser fragmentado, ou seja, dividido em dois blocos diferentes. Você não quer que isso aconteça, pois quer otimizar o tempo de acesso de seus arquivos.

Então, sempre que um arquivo for maior que o espaço disponível do bloco, você salvará ele no bloco seguinte.

Seu programa receberá o número de arquivos (N) que serão inseridos no HD (número inteiro).

Em seguida, o tamanho dos blocos do seu HD, que foi escolhido quando este foi formatado (também inteiro).

Por fim, você receberá o tamanho de cada um dos N arquivos que deverão ser salvos no HD.

Seu programa deverá retornar quantos blocos foram usados para armazenar todos estes arquivos no HD, na seguinte formatação: “Blocos ocupados: X’\n’”, no qual X é o número de blocos ocupados).

IMPORTANTE: A inserção dos arquivos deve ocorrer na ordem INVERSA da ordem de leitura dos dados (ou seja, o último arquivo recebido como entrada será o primeiro a ser inserido no HD, e o primeiro arquivo lido da entrada será inserido por último).

Obs1: Nenhum arquivo será maior que o tamanho do bloco.

Obs2: Lembre-se: toda vez que o arquivo for maior que o tamanho restante do bloco, ele deverá ser salvo no próximo bloco e o programa continuará salvando arquivos a partir deste novo bloco, ele não voltará para o anterior!

Obs3: A saída será a quantidade de blocos utilizados, cuidado com a manipulação do contador!

Obs4: O máximo de arquivos inseridos será 10.

Exemplo:

Entrada:

```
5
100
80
20
30
20
15
```

Saída:

```
Blocos ocupados: 2’\n’
```

Entrada:

```
6
200
200
150
75
```

115
25
100

Saída:

Blocos ocupados: 4’\n’

Exercício 3 (Valor: 4.0 pontos):

Você deverá verificar se um nome de usuário inserido no seu sistema é válido ou não. Dessa forma, 2 verificações são feitas:

1º o nome NÃO pode conter nenhum caractere especial, apenas letras (maiúsculas ou minúsculas) e números.

2º o nome DEVE conter ao menos uma letra minúscula, uma letra maiúscula E um algarismo numérico.

Cada verificação deve ser feita em uma FUNÇÃO diferente.

Cada função **receberá apenas a string** com o nome de usuário e **retornará um valor** que indique se a condição foi cumprida ou não.

Seu programa deverá imprimir "Username OK'\n'" se ambas as condições forem satisfeitas. A impressão **deve** ser feita na *main*.

Caso o nome de usuário possua algum problema, seu programa deverá imprimir, primeiramente: "Username invalido'\n'" (Sem acentos!).

Em seguida, você deverá imprimir qual o erro encontrado.

Se o erro foi um caractere especial encontrado, imprima: "Possui caractere especial'\n'".

Se o erro foi que não foram encontrados os 3 tipos de caracteres diferentes, imprima: "Nao possui os 3 tipos de caracteres'\n'" (Sem acentos!).

Obs1: Caso ocorram os dois erros no mesmo *username*, primeiro imprima a mensagem de erro do caractere especial, e depois o erro de não possuir os 3 tipos de caractere obrigatórios.

Obs2: O seu retorno pode ser um inteiro, em que 0 significa que a condição não ocorreu, e 1 significa que a condição ocorreu.

Obs3: Lembre-se: na tabela **ASC II**, as letras de 'a' a 'z' são **consecutivas**, as letras de 'A' a 'Z' também. O mesmo se aplica aos algarismos de '0' a '9'.

Obs4: O nome de usuário não terá mais de 30 caracteres.

Exemplos:

Entrada:

godofredo

Saída:

Username invalido'\n'
Nao possui os 3 tipos de caracteres'\n'

Entrada:

l33T

Saída:

Username OK'\n'

Entrada:

_hacker

Saída:

Username invalido'\n'
Possui caractere especial'\n'
Nao possui os 3 tipos de caracteres'\n'