Decifrando Relatórios

Programação de Computadores 1

Prof. Daniel Saad Nogueira Nunes

1 Contextualização

Inúmeros relatórios envolvendo transações ilegais foram encontradas pela polícia federal na sua mais recente operação: “Gizé". Cada transação é composta por uma soma de dois números, resultando em um terceiro. Contudo, os relatórios estão cifrados. Cada dígito foi substituído por uma letra maiúscula, isto é, os números se tornaram palavras.

Para resolver este problema, os coordenadores da operação resolveram utilizar a seguinte estratégia. Para cada transação, foi assumido que o valor resultante da soma é a maior possível, já que o objetivo é identificar a ilegalidade. Além disso, cada letra maiúscula corresponde a um algarismo, podendo letras diferentes apontarem para o mesmo algarismo. Obviamente, os números gerados a partir da correspondência, não podem conter zeros à esquerda, mas podem ser iguais a 0.

Como a pilha de relatórios só cresce, os coordenadores procuraram a sua ajuda para resolver o problema.

2 Especificação

O programa deverá receber o caminho de dois arquivos: um arquivo de entrada e um de saída. Os caminhos dos arquivos de entrada e saída deverão ser obtidos via argumentos de linha de comando:

- Primeiro argumento: caminho do arquivo de entrada.
- Segundo argumento: caminho do arquivo de saída.

2.1 Arquivo de entrada

O arquivo de entrada entrada possui três linhas com as palavras A, B e C, as quais descrevem uma transação, isto é, C equivale à soma de A e B. É garantido que as palavras não ultrapassam 8 caracteres e que o número de caracteres distintos nas três palavras é no máximo 7.

Restrições:

- $1 \leq |A|, |B|, |C| \leq 8$.

2.2 Arquivo de saída

A primeira linha do arquivo de saída deve possuir o maior valor possível de C. Caso não seja possível decifrar a transação, o valor a ser impresso é “-1”, senão, as próximas linhas devem descrever a correspondência utilizada para obter os números da transação. Cada uma destas linhas está no formato“$c : d$” em que c corresponde a uma letra e d corresponde a um algarismo.

No caso em que seja possível decifrar o relatório, as correspondências podem ser dadas em qualquer ordem.

2.3 Modularização

O sistema deverá ser dividido em módulos, cada qual com uma tarefa. Estes módulos podem ser organizados internamente através de várias funções e eles correspondem aos seguintes:

- Entrada: lê os relatórios de um arquivo texto.
- Saída: imprime o resultado da decifração em um arquivo texto.
- Processamento: decifra os relatórios, caso possível.
- Tratamento de erros: captura erros e reporta ao usuário, caso necessário.

Módulos adicionais também podem ser criados, a critério do programador. Os módulos devem ser organizados em arquivos separados, com seus respectivos arquivos de cabeçalho e implementação.

2.4 Construção do sistema

Um `Makefile` deverá ser produzido para a compilação dos códigos-fontes no executável e deverá ser distribuído junto ao código.

2.5 Documentação

O código deve ser bem documentado, com presença de comentários explicando os trechos mais complexos do código. Além disso, um arquivo `README.md` deve ser providenciado com a devida identificação do autor descrevendo o projeto e instruindo como o código deve ser compilado através da ferramenta `make`.

2.6 Exemplos

Entrada:

```
AB
A
BCC
```

Saída:

```
100
A:9
B:1
C:0
```

Entrada:

```
AB
A
CDC
```

Saída:

```
101
A:9
B:2
C:1
D:0
```

Entrada:

```
AB
BC
AC
```

Saída:

```
-1
```

Entrada:

```
PEAR
APPLE
GRAPE
```

Saída:

```
90852
A:8
E:2
G:9
L:7
P:5
R:0
```

3 Critérios de correção

Deve ser utilizada a linguagem de programação `C` para a implementação do interpretador.

Para validação da correção do algoritmo, testes automatizados serão realizados, então é **crucial** que a saída esteja conforme o especificado.

Serão descontados pontos dos códigos que não possuírem indentação.

Os códigos deverão ser documentados e obrigatoriamente devem possuir a presença de um arquivo `Makefile`.

3.1 Ambiente de Correção

Para a correção dos projetos, será utilizada uma máquina de 64-bits com sistema operacional `GNU/LINUX` e compilador `GCC 10.2.0`, logo é imprescindível que o sistema seja capaz de ser compilado e executado nesta configuração.

4 Considerações

- `GDB`, `Valgrind` e ferramentas gráficas associadas podem ajudar na depuração do código.
- Este projeto deve ser executado **individualmente**.
- Não serão avaliados códigos que não compilem ou sem a presença de um `Makefile`.
- Como a correção é automatizada, deverá ser impresso apenas o que a especificação pede. Atente-se para a formatação da saída.
- A incidência de plágio será avaliada automaticamente com nota 0 para os envolvidos. Medidas disciplinares também serão tomadas.
- O trabalho deve ser entregue dentro de uma pasta zipada com a devida identificação do aluno no prazo combinado pelo ambiente virtual de aprendizagem da disciplina.