UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br/

Professor: Rodrigo Fernandes de Mello (mello@icmc.usp.br)
Alunos PAE: Lucas Pagliosa (lucas.pagliosa@usp.br)
Felipe Duarte (fgduarte@icmc.usp.br)
Monitores: Victor Forbes (victor.forbes@usp.br)

Trabalho 06: Anagramas

1 Prazos e Especificações

O trabalho descrito a seguir é individual e não será tolerado qualquer tipo de plágio ou cópia em partes ou totalidade do código. Caso seja detectada alguma irregularidade, os envolvidos serão chamados para conversar com o professor responsável pela disciplina.

A entrega deverá ser feita única e exclusivamente por meio do sistema run.codes no endereço eletrônico `https://run.codes` até o dia **16 de Novembro de 2016 às 23 horas e 59 minutos**. Sejam responsáveis com o prazo final para entrega, o run.codes está programado para não aceitar submissões após este horário e não serão aceitas entregas fora do sistema.

Leia a descrição do trabalho com atenção e várias vezes, anotando os pontos principais e as possíveis formas de resolver o problema. Comece a trabalhar o quanto antes para que você não fique com dúvidas e que consiga entregar o trabalho a tempo.

As limitações computacionais do Run.codes (memória e tempo de processamento) fazem parte do escopo deste trabalho e devem ser levados em consideração no desenvolvimento de sua solução. Assim, elabore um algoritmo que utiliza pouca memória e executa cada caso de teste em menos de 1 segundo.

2 Descrição do Problema

Um anagrama é o rearranjo das letras de uma palavra produzindo outras palavras (utilizando todas as letras originais exatamente uma vez). Um exemplo conhecido é o nome da personagem "Iracema", claro anagrama de "América", no romance de José de Alencar.

Assim, o problema consiste em encontrar os maiores conjuntos de anagramas de um dicionário da língua inglesa com no máximo 300000 verbetes (Não existe no dicionário palavras com acentos gráficos e nem caracteres especiais). Caso exista mais de um grupo com a mesma quantidade de palavras, o seu programa deve ser capaz de encontrar todos.

Para ilustrar o problema considere um dicionário formado pelas palavras:

```
cornelia
bedlar
alcor
acrolein
calor
creat
belard
```

O seu programa deve ser capaz de identificar que o maior conjunto de anagramas têm duas palavras e, além disso, existe três subconjuntos com tamanho máximo, em nosso exemplo os subconjuntos {`acrolein, cornelia`}, {`alcor, calor`} e {`bedlar, belard`}.

O seu programa deverá solucionar o problema de maneira mais eficiente possível levando em consideração o baixo consumo de memória e alto desempenho. Assim, a partir da leitura, **utilize uma tabela Hash para solucionar o problema** de maneira eficiente. Lembre-se que na tabela ASCII, cada letra possui um número e eles podem ser utilizados para o cálculo de sua função hash. Para facilitar o desenvolvimento do trabalho, aconselha-se fortemente a leitura da Seção 3.4 do livro "Algorithms" – Robert Sedgewick e Kevin Wayne. Essa leitura não é obrigatória mas pode ajudar na resolução do problema.

3 Entrada e Saída

O seu programa receberá como entrada somente o nome do arquivo que contém as palavras do dicionário. Novamente, esse arquivo deverá ser aberto uma única vez e suas palavras lidas apenas uma única vez, ou seja, não serão aceitos trabalhos que abram mais de uma vez o arquivo ou que manipulem o ponteiro do arquivo para ler uma palavra mais de uma vez. Um exemplo de entrada do sistema é:

```
01.txt
```

Após encontrar todos os conjuntos de anagramas, o seu programa deverá ser capaz de informar o número de conjuntos que possuem a maior quantidade de anagramas do dicionário, ou seja, se o maior grupo conter 4 palavras e existir 3 grupos com tamanho 4, o seu programa deverá imprimir o valor 3. Em seguida, deverá imprimir todas as palavras de cada conjunto (1 conjunto por linha) separando as palavras por virgula.

Para que seja possível comparar a saída do seu programa com a saída esperada pelo run.codes, as palavras devem ser impressas em ordem alfabética dentro do seu conjunto e, por sua vez, os conjuntos devem ser impressos em ordem alfabética de acordo com a primeira palavra do conjunto. Um exemplo de saída esperada do sistema é:

```
6
acinar, arnica, canari, carian, carina, crania
acrolein, arecolin, caroline, colinear, cornelia, creolian
albeit, albite, baltei, belait, betail, bletia
alcor, calor, carlo, carol, claro, coral
balder, bardel, bedlar, bedral, belard, blader
caret, carte, cater, crate, creat, creta
```

Observe que entre as palavras deve existir uma virgula seguida de uma espaço em branco e, após a ultima palavra, deve existir uma quebra de linha.

4 Dicionário

O arquivo que contém as palavras do dicionário possui uma palavra por linha e não está em ordem alfabética. Um exemplo de arquivo seria:

```
cornelia
bedlar
alcor
acrolein
calor
creat
belard
```

5 Observações importantes

- Programe as impressões na tela EXATAMENTE como exemplificado no decorrer deste documento. Tome cuidado com pulos de linha, tabs, espaços, etc.
- Coloque dentro do zip todos os arquivos de código (`*.h *.c`), o `Makefile` e um arquivo `README` com o nome e número USP.
- Para resolução deste trabalho aconselha-se fortemente a leitura da Seção 3.4 do livro "Algorithms" – Robert Sedgewick e Kevin Wayne
- Os recursos computacionais do Run.codes são limitados e essa restrição faz parte do problema. Elabore uma solução que consuma pouca memória e que forneça a resposta em no máximo 1seg. por caso de teste.
- O dicionário fornecido contém palavras da língua inglesa (não há acentos nem caracteres especiais) com todos os caracteres minúsculos.