ICC2 – Trabalho 1

Utilizando a linguagem C, considere um arquivo texto cujo nome será encaminhado como parâmetro para seu programa. Leia o nome desse arquivo utilizando a função scanf. A seguir, crie um ponteiro para armazenar todo o conteúdo desse arquivo em memória Heap (http://pt.wikipedia.org/wiki/Gerenciamento_de_mem%C3%B3ria).

Faça então a leitura do arquivo texto, um caracter por vez, e utilize a função realloc para realizar a alocação da memória necessária, até armazenar todo o conteúdo do arquivo texto em memória Heap. Assim, você terá uma região de memória dinamicamente alocada para guardar todo o conteúdo do arquivo.

Em seguida, seu programa receberá como entrada instruções para realizar ou não cada uma das 5 tarefas abaixo, gerando como saída (imprimindo na tela) os resultados. A entrada consiste de sete 0s e 1s (zeros e uns), em que:

1. O dígito 1 significa que a tarefa deverá ser executada, e
2. O dígito 0 significa que a tarefa não deverá ser executada.

Por exemplo, a entrada abaixo:

```
10001
```

Significa que o programa deverá executar apenas as Tarefas 1 e 5

As Tarefas são:

1) Conte e imprima o número total de palavras no arquivo. Observação: qualquer palavra deve ser contada, inclusive preposições, artigos, etc.;

2) Conte e imprima o número total de ocorrências das seguintes palavras, especificamente:

a, ante, até, após, com, contra, de, desde, em, entre, para, per, perante, por, sem, sob, sobre, trás

3) Conte e imprima o número de espaços.

4) Conte e imprima o número de tabulações (tabs) existentes no arquivo.

5) Conte e imprima o número de palavras que possuam mais de 10 caracteres.

Formato da entrada:

```
<nome_do_arquivo>
<T1><T2><T3><T4><T5>
```

Formato da saída:

```
<numero_inteiro_item_Ti>\n
...
```

<numero_inteiro_item_TN>\n

Atenção: a saída NÃO DEVE ser impressa para Tarefas não solicitadas!

Exemplo de Entrada e Saída:

```
Arquivo.txt
--------------------------------------------------------------------------------
João joga futebol com o pessoal da escola desde o ano passado. Ele é um aluno
muito dedicado
--------------------------------------------------------------------------------
```

Entrada fornecida:

```
Arquivo.txt
11111
```

Saída esperada:

```
18
2
17
0
0
```

Entrada fornecida:

```
Arquivo.txt
10110
```

Saída esperada:

```
18
17
0
```

Informações importantes (LEIA COM ATENÇÃO)

- Sobre a avaliação

1. Um dos objetivos da disciplina de ICC2 é o **aprendizado individual** dos conceitos de programação. A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos, nem codificar em conjunto. Portanto, compartilhem idéias, soluções, modos de resolver o problema, mas não o código.
    - O plágio vai contra o código de ética da USP.
    - Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação.
    - O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e idéiar. Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que para poupar memória basta usar uma certa estrutura de dados, etc. O que você **não** deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado do seu colega:
        - depois de o seu colega ter visto o seu código, será muito mais difícil para ele imaginar

uma solução original e própria;
    - o seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.
  - Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela.
  - O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2. Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e **os trabalhos com alta similaridade detectada terão suas notas zeradas**, tanto aqueles relativos ao código de origem quanto do código copiado. **A detecção de plágio será reportada à Seção de Graduação para providências administrativas**.

3. A avaliação incluirá a porcentagem de acertos verificada pelo SQTPM e também a análise do seu código, incluindo endentação, comentários, bom uso da memória e práticas de programação. Portanto faça seu código com cuidado, da melhor forma possível.

- Sobre o sistema de submissão:

1. Seu código deverá incluir arquivo fonte .c .h e Makefile – todos os arquivos deverão **obrigatoriamente** conter no início um comentário com seu nome, número USP, turma e data da entrega do trabalho.

2. **A data/hora de entrega do trabalho é aquela estipulada no sistema. Trabalhos entregues por email NÃO serão aceitos, mesmo que dentro da data/hora estipulada. Faça seu trabalho com antecedência para evitar entregar em cima da hora e ter problemas de submissão, pois o sistema tende a ficar lento com as múltiplas submissões feitas geralmente próximas ao fechamento do sistema.**
    - A submissão é de responsabilidade do aluno, e os problemas comuns à entrega próxima ao fechamento do sistema também. Portanto: problemas de acesso à rede **não** serão aceitos como desculpa para entrega por email ou fora do prazo.

3. A compilação e execução do código é feita no sistema pelos comandos:
```
make all
make run
```

4. A saída do seu programa deve ser exatamente igual à saída esperada, incluindo: espaços em branco, quebras de linha e precisão decimal.

5. Há um limite em segundos para a execução dos casos de teste e um limite de memória total para ser utilizada. Você deverá gerenciar bem o tempo de execução e o espaço ocupado para que seu programa fique dentro desses limites, para evitar uso excessivo, pois o sistema irá invalidar o caso em que o limite foi excedido.

6. Ao enviar, aguarde **sem recarregar a página nem pressionar ESC**, para obter a resposta. Caso demore mais do que 2 minutos para dar uma resposta, feche e abra novamente a página do servidor. Verifique se seu programa tem algum problema de entrada de dados (ou se tem algum loop infinito ou parada para pressionamento de tecla). Caso não tenha, aguarde alguns minutos e tente novamente.

7. O erro de “Violação de Memória” significa acesso indevido a arranjo ou arquivo. Use compilação com **-g** e o programa valgrind para tentar detectar a linha de código com erro.

   - Exemplo 1 de violação de memória:

```
int **mat = (int **)malloc(sizeof(int *) * 3);
mat[0] = (int *)malloc(sizeof(int)*10);
for (i = 1; i < 3; i++) {
   free(mat[i]);
}
// apenas a posicao 0 de mat foi alocada as outras nao
// portanto esta liberando regiao nao alocada
// gerando violacao de memoria
```

   - Exemplo 2 de violação de memória:

```
int B[5] = {5, 6, 7, 8, 9};
int N = 5;
int *A = malloc(N*sizeof(int));
int j = 0, i = 1; // 'j' inicializado em 0, 'i' inicializado em 1
while (j < N){
      A[i] = B[j]; // 'j' e 'i' sao indices diferentes
      j++;         // quando j = (N-1) i = (N-1)+1 e
      i++;             // havera escrita indevida em A
}
```