Tarefa 1

Formato de entrega:

Entregar todos os programas em C e uma descrição contendo teste de mesa detalhado.

Local de entrega: depositar no Moodle de EDA-TADS

Equipes: máximo três elementos

I) Implementação sobre Pilhas: verificador de tags html

O trabalho consiste na implementação de um programa verificador de estrutura HTML simples utilizando a estrutura de dados pilha. O programa deve ser capaz de verificar se um determinado arquivo HTML está estruturado corretamente, ou seja, se todas as tags html estão abertas e fechadas numa ordem válida.

Exemplo de arquivo HTML válido:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Title of the document
        </title>
    </head>
    <body>
        The content of the document.....
    </body>
</html>
```

Exemplo de arquivo HTML inválido:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Title of the document
        </title>
        <body>
    </head>
        The content of the document...
        </body>
</html>
```

No exemplo acima a tag mais externa “head” é fechada antes da sua tag aninhada “body” ser , isto caracteriza um erro. Em um arquivo HTML, tags são abertas segundo o padrão menor-que, nome-da-tag e maior-que (<nome_da_tag>) e fechadas segundo o padrão menor-que, barra, nome-da-tag e maior-que (</nome_da_tag>). Existem algumas tags que não necessitam de fechamento, por exemplo, a tag “!DOCTYPE” não necessita fechamento.

Definição do programa:

• A implementação do programa verificador utiliza uma Pilha implementada em C (utilize uma pilha disponível no Moodle da disciplina);

• O nome do arquivo a ser verificado pelo programa deve ser passado como parâmetro da linha de comando. Dica: utilize argc e argv;

Ex. >>> ./verificador-html arquivo.html

• Caso não haja erro algum no arquivo o programa deve sair sem apresentar mensagem alguma. Em outro caso, deve ser impresso na tela uma mensagem semelhante a “ERRO: esperado '</body>', recebido '</head>'” e o programa deve encerrar a execução;

• A comparação entre tags é feita pelo seu nome. As tags estão contidas entre um símbolo '<' (menor-que) e um símbolo '>' (maior-que), o nome da tag é a primeira string contida após '<' até o primeiro espaço. ex. Para a tag '<input type="text" name="fname">' o nome da tag é 'input';

• O nome de uma tag é formada de no máximo cinquenta (50) caracteres;

• A tags que não necessitam fechamento são apresentadas na tabela abaixo: Tags que não necessitam fechamento <!DOCTYPE>, <input>, <frame>,
, <img>

• Não é necessário tratar a tag de comentário “<!-- ... -->”

Pseudo código:

```
Abre arquivo fonte html
Cria pilha
Lê primeiro nome-tag e o empilha
        ENQUANTO houver tags no arquivo
                Lê próxima tag
                SE nome-tag é precedido uma barra
                        SE pilha contém no topo o mesmo nome-tag sem a barra
                                Desempilha o topo;
                        SENAO
                                ERRO de “aninhamento” da nome-tag na linha xx
                                Encerra
                SENAO
                        Empilha nome-tag
SE pilha não vazia
                ERRO de “aninhamento” da nome-tag na linha xx
                Encerra
SENAO
        Html com as tags bem “aninhadas”
```

EXEMPLO 1:

ERRO detectado em t_3: a tag *head* é fechada, porém o último bloco aberto é do tipo *title*

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>
5.              Title of the document
6.      </head>
7.          </title>
11. </html>
```

Instante	t_o	t_1	t_2	t_3
Próxima tag	<html>	<head>	<title>	</head>
PILHA	<html>	<head> <html>	<title> <head> <html>	<title> <head> <html>
Status Pilha	Insere tag	Insere tag	Insere tag	Insere tag

EXEMPLO 2:

SEM ERRO de “aninhamento” de tags.

```
1.  <!DOCTYPE html>
2.  <html>
3.      <body>
4.      The content of
        the document
5.      </body>
6.  </html>
```

Instante	t_o	t_1	t_2	t_3	t_4	t_5
Próxima tag	<html>	<body>	<\body>		</html>	
PILHA	<html>	<body> <html>	~~<body>~~ <html>	<html>	~~<html>~~	
Status Pilha	Insere tag	Insere tag	Remove tag		Remove tag	vazia

II) Conta linhas/colunas

Implemente uma função (apenas a função) e realize teste-de-mesa (execução do algoritmo em papel para as situações possíveis a serem tratadas pelo algoritmo). A função determina as dimensões de uma estrutura matricial, similar àquela exibida na Fig. 1, e retorna uma struct contendo o total de linhas e de colunas da matriz. Observe que o ponteiro 'p' é a ligação para o primeiro nó da estrutura matricial. Se a estrutura estiver vazia `p` será igual a null.

Protótipo da função *struct dim contaLC(struct nodo *pt)*

```
struct dim{
    int nlinhas;
    int nColunas;
};
```

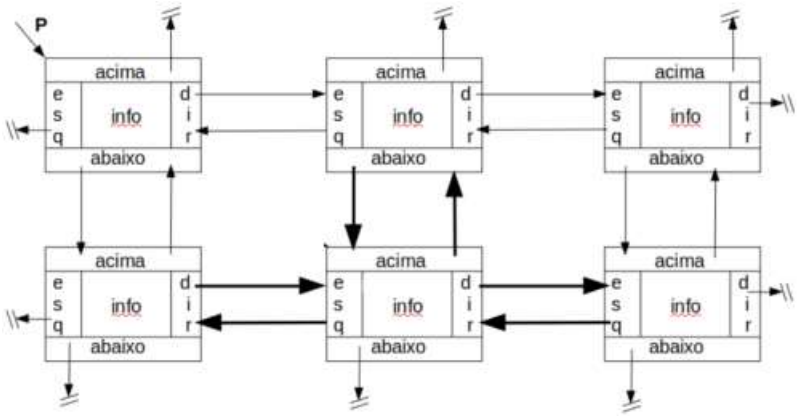

Figura 1 – Um exemplo da estrutura matricial.