MAC211 – Laboratório de Programação I - 2013

Exercícios para a Prova 2
(Algumas Respostas)

Tema: Expressões Regulares

1. Escreva uma expressão regular (usando a sintaxe das expressões estendidas) que reconheça:
1.1 cadeias que possuam somente letras maiúsculas e minúsculas alternadas.

[[A-Z][a-z]]*[A-Z]?|[[a-z][A-Z]]*[a-z]?

1.2 cadeias que tenha todas as vogais pelo menos uma vez e em ordem.

.*[Aa].*[Ee].*[Ii].*[Oo].*[Uu].*

1.3 todos os números inteiros positivos menores que 3000.

(3000|[1-2]?[0-9]?[0-9]?[0-9])

1.4 todos os números hexadecimais que podem ser representados com um byte.

1 dígito hexadecimal [0-9A-F] = 4 bits. Logo, 1 byte corresponde à 2 dígitos hexadecimais

[0-9A-Fa-f]?[0-9A-Fa-f]

1.5 exatamente 100 cadeias diferentes.

[0-9][0-9] ou [A-J][k-t] ou [A]{1,100}

Obs: existem várias respostas possíveis!

1.6 exatamente 101 cadeias diferentes.

([0-9][0-9]|100) ou ([A-J][k-t]|u) ou [A]{1,101}

Obs: existem várias respostas possíveis!

Tema: Awk

1. Escreva um script em awk que lê um arquivo da entrada padrão e gera uma tabela de todas as palavras de comprimento maior do que 3 caracteres, onde cada linha contém a palavra na primeira posição, seguida por um ':' seguido pelos números das linhas onde a palavra ocorre. Por palavra, entenda uma sequência de letras. Depois, responda as perguntas abaixo (justificando suas respostas):

```
        {
                for (i = 1; i <= NF; i++) {
                        if ( $i ~ /.*[A-Za-z][A-Za-z][A-Za-z].*/ ) {
                                freq[$i]++
                                linhas[$i,freq[$i]] = NR
                        }
                }
        }

END     {
                for (palavra in freq) {
                        printf "%s: ", palavra
                        for (i = 1; i <= freq[palavra]; i++)
                                printf "%d ", linhas[palavra,i]

                        printf "\n"
                }
        }
```

1.1 As linhas aparecerão em ordem?

Sim, porque para vetor que armazena os números das linhas de cada palavra, usamos um índice numérico incremental. Como as linhas do arquivo de entrada são processadas ordenadamente, do início ao fim do arquivo, isso garante que os números de linhas exibidos para cada palavra aparecerão na ordem correta.

1.2 As palavras aparecerão em ordem?

Não necessariamente. Para o vetor de palavras, usamos um índice que é uma string. Não é possível garantir que os itens estarão armazenadas em posições contíguas do vetor.

2. Para cada uma das chamadas do awk abaixo, explique o que ela faz.
2.1 awk 'length($0) > 80'

Exibe as linhas do arquivo de entrada que possuem mais do que 80 caracteres.

2.2 awk '{ if (NF > 0) print }'

Exibe as linhas não vazias do arquivo de entrada.
2.3 awk '{ printf “%d: %s\n”, NR, $0 }'

Exibe todas as linhas do arquivo de entrada, acrescentando na frente de cada linha o número da linha seguido por ': '

2.4 awk '/banana/ { print $2 $1 }'

Para toda linha que contém a string “banana”, exibe o segundo campo da linha seguido pelo primeiro campo.

Tema: Flex

1. Escreva uma especificação em flex para um filtro que copia um texto da entrada para a saída, trocando:

- todas as sequências consecutivas de espaços e TABs por um único espaço,
- todos os espaços em branco nos finais das linhas por nada (remoção) e
- todas as ocorrências da palavra *usuário* pelo nome do usuário devolvido pela função char* getlogin(). Para usar essa função, é preciso incluir o arquivo de cabeçalho unistd.h.

```
%{
  #include <unistd.h>
%}

%%
[ \t]+       { printf(" ");  }
[ ]+\n       { printf("\n"); }
"usuário"    { printf("%s", getlogin()); }

%%

int main(int argc,char** argv){
        yylex();                /* executa o scanner  */
}
```

Tema: Bison

1. Escreva um scanner em flex + um parser em bison para processar um arquivo contendo dados de ex-alunos do IME que possui o seguinte formato esperado:

MAC	3132425	Ana Maria Machado	MAC123-4 MAT325-6 MAC211-4
MAT	2128902	Carlos Drummond de Andrade	MAT325-8
MAE	2871218	Cecilia Meirelles	MAE565-6 MAC122-4 MAC211-4 MAE258-6
MAE	2329803	Fernando Pessoa	MAC211-4 MAE258-6

Nesse arquivo, cada linha contém os dados de um aluno. Os campos estão separados por uma ou mais tabulações. O primeiro campo contém a sigla do departamento do aluno (que pode ser MAC, MAT ou MAE); o segundo contém o número USP do aluno; o terceiro, o nome; e o quarto é a lista dos códigos das disciplinas cursadas pelo aluno (separados por espaços). O código de uma disciplina é composto pela sigla do departamento que a ofereceu, seu número e, separado por um '-', o número de créditos da disciplina (que está entre 1 e 9).
O seu parser deve mostrar, para cada aluno, o total de créditos cursados. E, no final do processamento do arquivo, deve mostrar o número de alunos contabilizados em cada um dos 3 departamentos.

MAC0211

```
/*  Arquivo: parser_alunos.l
    Entrada para o Bison        */

%{
  #include <stdio.h>
  #include <string.h>
  int cont_MAC = 0, cont_MAT = 0, cont_MAE = 0;
%}

%union{
 char* string;
 int   val;
}

%token <string> DEPARTAMENTO NUSP NOME
%token <val>    COD_DISCIPLINA
%type  <val>    aluno disciplinas

%%

arquivo    : /* vazio */
             | arquivo linha;

linha      : '\n'
             | aluno '\n';

aluno      : DEPARTAMENTO '\t' NUSP '\t' NOME '\t' disciplinas
             {
               if (strcmp($1, "MAC") == 0)
                 cont_MAC++;
               else if (strcmp($1, "MAT") == 0)
                 cont_MAT++;
               else
                 cont_MAE++;
```

```
            $$ = $7;
            printf("Aluno: %s -- Total de creditos: %d \n", $5, $$);
          };

disciplinas  : COD_DISCIPLINA                  { $$ = $1; }
             | disciplinas ' ' COD_DISCIPLINA  { $$ = $1 + $3; } ;

%%

int main(){
  yyparse();
  printf("A quantidade de alunos por departamento e': \n");
  printf(">>> MAC = %d\n", cont_MAC);
  printf(">>> MAT = %d\n", cont_MAT);
  printf(">>> MAE = %d\n", cont_MAE);
}

yyerror (char * s) {
  printf ("Arquivo de alunos invalido: %s\n", s);
}
```

```
/*  Arquivo: scanner_alunos.l
    Entrada para o Flex       */

%{
  #include "parser_alunos.tab.h"
%}

DEPTO MAC|MAT|MAE

%%

[ \t]*[\t][ \t]*          { return '\t';  /* separador de campos */                                  }
{DEPTO}                   { yylval.string = strdup(yytext); return DEPARTAMENTO;                   }
{DEPTO}[0-9]+[-][0-9]     { yylval.val = atoi(&yytext[strlen(yytext)-1]); return COD_DISCIPLINA; }
[0-9]+                    { yylval.string = strdup(yytext); return NUSP ;                          }
[A-Z][A-Za-z ]*[a-z]      { yylval.string = strdup(yytext); return NOME ;                          }
[ ]*                      { return ' '; /* separador de disciplinas */                             }
\n|.                      { return yytext[0];                                                      }

%%
```