TEMA – CERINTE

1. Implementati problemele rezolvate din MATERIAL.

2. Realizati un scanner care inlocuieste toate nr intregi dintr-un text cu corespondentele lor in baza 2. (1 punct)

3. Asupra unui text de intrare, realizati urmatoarea transformare: fiecare vocal/grup de vocale va fi precedat de grupul de litere **av**. De exemplu, pentru *iar la scoala* se va obtine ***av**iar l**av**a sc**av**oal**av**a*. (1 punct)

4. Realizati un scanner care recunoaste intr-un program C constantele intregi (in baza 10, 8, 16 si cele de tip intreg derivat: unsigned, etc.), constantele reale (simpla si dubla precizie, inclusiv cele in format stiintific si cele de tip derivat – long double, etc.), constantele caracter si cele sir de caractere. (2 puncte)

5. Prezentarea pe web a unei portiuni de cod C/C++ (3 puncte)
   Intrare: fis. Sursa C/C++
   Iesire: fisier HTML care respecta formatarea din sursa C (indentare, linii noi) si evidentiaza cuvintele cheie, instructiunile preprocesor, sirurile de caractere si comentariile.
   De ex., fie programul sursa:

```
#include <fifo.h>
#define NVOISINS 8
for (x = 0; x < N; x++) {
  if (M[x] != 0)                    /* le pixel appartient a un maximum */
    {
    for (k = 0; k < NVOISINS; k += 1) /* parcourt les voisins */
      {                               /* si un voisin n'est pas dans la FIFO */
      y = voisin(x, k, rs, N);      /* et pas maximum, on le met en FIFO */
      if ((y != -1) && (! IsSet(y, EN_FIFO)) && (M[y] == 0)) {
        FifoPush(FIFO, y);
        Set(y, EN_FIFO);
        if (trace) printf("empile point %d (%d,%d)\n", y, y%rs, y/rs);
        } /* if y ... */
      } /* for k */
    } /* if M */
  } /* for x */
```

   Rezultatul (o parte):

```
<html><body><pre>
<span class="prepro">#include &lt;fifo.h&gt;</span>
<span class="prepro">#define NVOISINS 8</span><span class="kw">for</span> (x = 0; x &lt; N; x++)
  {  <span class="kw">if</span> (M[x] != 0)                    <span class="comment">/* le pixel appartient a un maximum */
    <span class="kw">for</span> (k = 0; k &lt; NVOISINS; k += 1) <span class="comment">/* parcourt les voisins */</span>
      {                            <span class="comment">/* si un voisin n'est pas dans la FIFO */</span>
      y = voisin(x, k, rs, N);     <span class="comment">/* et pas maximum, on le met en FIFO */</span>
      <span class="kw">if</span> ((y != -1) &amp;&amp; (! IsSet(y, EN_FIFO)) &amp;&amp; (M[y] == 0))
        {
        FifoPush(FIFO, y);
        Set(y, EN_FIFO);
        <span class="kw">if</span> (trace) printf(<span class="str">"empile point %d (%d,%d)\n"</span>, y, y%rs, y/rs);
        } <span class="comment">/* if y ... */</span>
      } <span class="comment">/* for k */</span>
    } <span class="comment">/* if M */</span>
  } <span class="comment">/* for x */</span>
</pre></body></html>
```

Pentru a obtine, de exemplu, cuvintele cheie ingrosate, sirurile de caractere in verde si comentariile in rosu, se poate utiliza fisierul CSS urmator:

```
.kw { font-weight: bold; }
.str { color: green; }
.comment { color: red; }
```

Lista cuvintelor cheie luate in considerare:

asm	auto	break	catch	case	char
class	const	continue	default	delete	do
double	else	enum	extern	float	for
friend	goto	if	inline	int	long
new	operator	overload	private	protected	public
register	return	short	signed	sizeof	static
struct	switch	this	template	typedef	union
unsigned	virtual	void	volatile	while	

6. Realizati un analizor lexical (cu ajutorul flex-ului) pentru limbajul C- (o submultime a lbj C++). (3 puncte).

Unitatile lexicale ptr C- sunt:

- Cuvinte cheie: **cin cout const else if typedef while**
- Simboluri: **+ - * / % ! || && < <= > >= == != << >> & | = ( ) [ ] { } , ‘ ;**
- Identificatori: **litera(litera | cifra)***
- Constante intregi: **cifra (cifra)***
- Const. car: **‘x’** sau **‘\n’**
- Const. reale (ca in C++, inclusiv format stiintific)
- Nume tipuri: char int float
- Identificatori predefiniti: main void
- Trebuie ignorate: spatiu, linie noua, comentarii (intr /* si */)

Exemplu de codificare a cuvintelor cheie si a simbolurilor:

symbole	code	symbole	code	symbole	code
+	1	{	23	«	45
-	2	}	24	»	46
*	3	[	25	||	47
/	4	]	26	&&	48
if	100	for	112	while	124

Cand analizorul recunoaste in fluxul de intrare unul dintre aceste simboluri, va afisa pe ecran codul corespunzator acestuia.

Pentru **constantele numerice**:
Analizorul va afisa:

- **Un cod semnificand (fie const. intreaga, fie reala, fie const. caracter) SI valoarea lexicala a acestei constante**

Identificatorii:
Pe masura ce acestia apar in fisierul de intrare, vor fi introdusi in tabela de simboluri. Tabela de simboluri stabileste corespondenta intre un identificator si un intreg (indexul identificatorului in tabela). Unii identificatorii (main, char, int, float, void) sunt predefiniti: fac parte din limbaj si trebuie introdusi in tab de simb la initializarea analizorului lexical.

In cazul identificatorilor, analizorul va afisa:

- **Un cod numeric semnificand: identificator**
- **Un intreg corespunzand indexului acestui identificator in tab de simboluri**

Daca acelasi identificator apare de mai multe ori in program, va fi inregistrat in tabela de simb. o singura data!!!!

Tabela de simboluri

Implementare: vector de siruri de caractere, tabele de dispersie, etc.

Exemplu:

Pentru codul:

```
void segmente()
{
  int mesure;
  int increment = 1;
  int maximise = 0;

  mesure = 0;
...
```

Tabela de simboluri:

index	chaine
0	segmente
1	mesure
2	increment
3	maximise

Veti implementa:

- Functie de cautare a unui sir de caractere in tabela
- Functie de adaugarea a unui sir in tabela.

Fiecare student va trimite o arhiva cu numele ***Tema1_NumePrenume***. (exemplu: Tema1_IonescuBogdan.rar (.rar, .zip., ce doriti)). **Va rog sa respectati numele arhivei!!!!**

Fiecare arhiva va contine in **6 directoare distincte** (P1, P2, …. , P6).