Tarea 1

Compiladores
Facultad de Ciencias, UNAM

1. Indica los valores asignados a w, x, y y z en los siguientes dos códigos estructurados por bloques. Muestra la tabla de símbolos en cada bloque con una implementación imperativa en cada caso:

```
int w, x, y, z;
int i = 4; int j = 5;
{
    int j = 7;
    i = 6;
    w = i + j;
}
x = i + j;
{
    int i = 8;
    y = i + j;
}
z = i + j;
```

```
int w, x, y, z;
int i = 3; int j = 4;
{
    int i = 5;
    w = i + j;
}
x = i + j;
{
    int j = 6;
    i = 7;
    y = i + j;
}
z = i + j;
```

2. Divide el siguiente programa en C++ en lexemas y genera los tokens correspondientes:

```
float limitedSquare(x) float x; {
    /* returns x-squared, but never more than 100 */
    return (x<=-10.0||x>=10.0)?100:x*x;
}
```

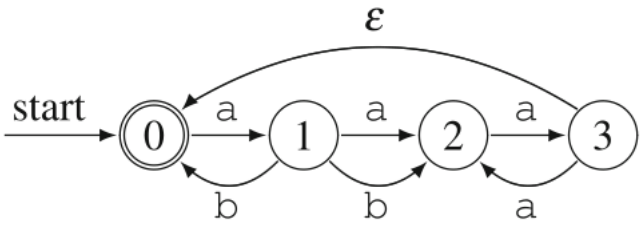
3. Define una función recursiva que compute los prefijos de una una expresión regular. La base de tal función recursiva es:

$$prefix(\epsilon) = \epsilon$$
$$prefix(a) = a?$$

   Completa la definición.

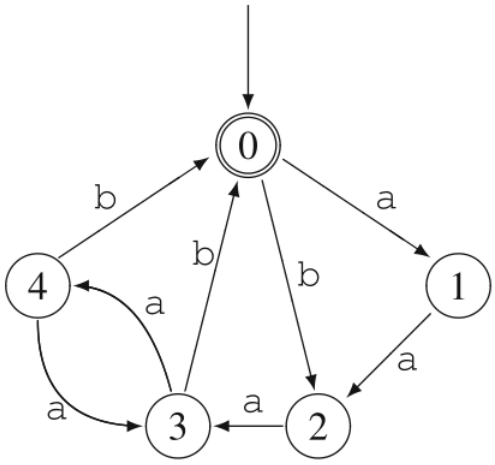
4. Para las siguientes expresiones regulares, da el lenguaje que definen:
   (a) $[ab][cd\epsilon]$
   (b) $[a - zA - Z]^* at^*$
   (c) $ca[tr]$

5. Para el siguiente autómata finito no determinista, construye el autómata finito determinista:

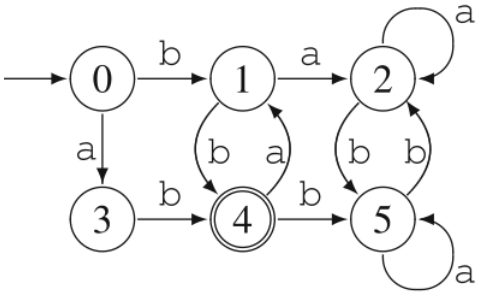


6. Para los siguientes DFA obten el DFA mínimo:

(a)



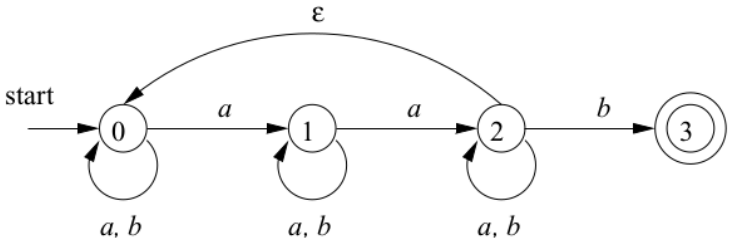
(b)



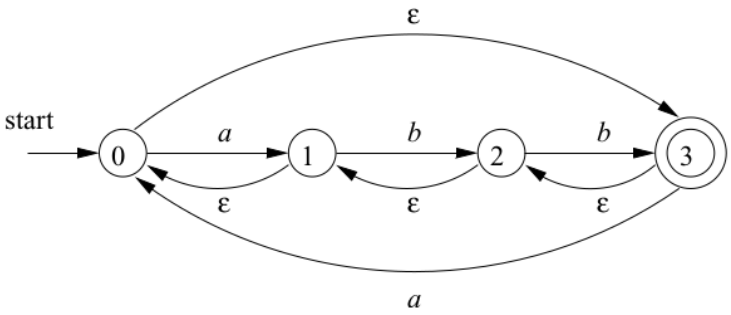
7. Convierte las siguientes expresiones regulares en autómatas finitos deterministas (DFA):
   (a) $[ab]^*$
   (b) $(a?b^*)^*$
   (c) $[ab]^*abb[ab]^*$

8. Utiliza el algoritmo de simulación de NFA para simular los siguientes NFAs en la entrada *aabb*:

(a)



(b)



9. Define un lexer para las expresiones booleanas que considere los siguientes puntos:
   (a) Que considere las constantes `True` y `False` como tokens `const`.
   (b) Que considere las vsriables (`var`) x, y, z, p, q, r, etc.

(c) Que considere los operadores binarios (`binop`) básicos: $\wedge$, $\vee$ y $\neg$.

(d) Define las regex de este para este lexer.

10. Del ejercicio anterior, utiliza las regex de las expresiones booleanas para definir el AF, comviértelo en AFD y redúcelo. Pruébalo con las siguientes expresiones para obtener los tókens:

    (a) $p \wedge q$

    (b) `True` $\wedge \neg (p \vee q)$

    (c) $\neg(x \wedge y) \vee (p \wedge q)$