پاسخ تکلیف تئوری سری اول

استاد درس: حسین فلسفین

دستیاران آموزشی: علی آهنگرپور – علیرضا صالحی حسین آبادی

۱

ظاهرشدن حرف‌های غیرمجاز(Appearance of illegal characters)

```
#include <stdio.h>

int main() {
    puts("string”);$
    return 0;
}
```

عدم تطبیق رشته(Unmatched string)

```
#include <stdio.h>

int main() {
    /* unfinished comment
    puts("string");
    return 0;
}
```

۱ (ادامه)

غلط املایی(Spelling Error)

```
#include <stdio.h>

int main() {
    float a = 123.g;
    return 0;
}
```

جایگزینی یک حرف با یک حرف نادرست(Replacing a character with an incorrect character.)

```
#include <stdio.h>

int main() {
    int a = 12f3;
    return 0;
}
```

۱ (ادامه)

- حذف حرفی که باید حضور داشته باشد.(Removal of the character that should be present.)

Original input: #include <istream>

After transposition: #include<iostream>

- جابجایی دو حرف(Transposition of two characters)

Original input: `retrun 0;`
After transposition: `return 0;`

- بیش از حد شدن طول شناسه‌ها یا ثابت‌های عددی(Exceeding length of identifier or numeric constants)

```
int a=2147483647 +1;
```

۲

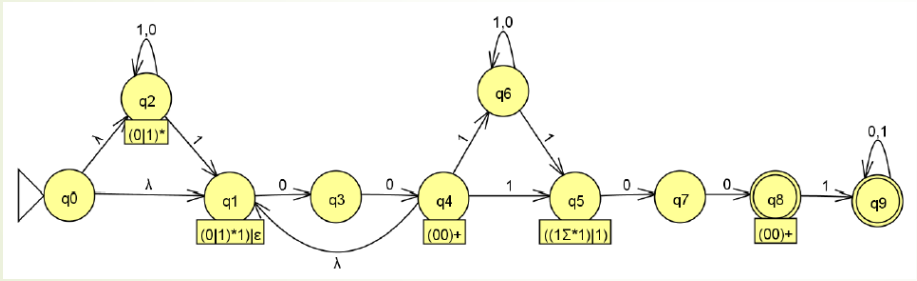
- کامپایل در مقابل تفسیر: یک کامپایلر کل کد منبع یک برنامه را قبل از اجرای آن به کد ماشین ترجمه می‌کند، درحالی که یک مفسر کد منبع را یک خط در یک زمان ترجمه و اجرا می‌کند. این بدان معنی است که یک برنامه کامپایل شده سریعتر از یک برنامه تفسیر شده اجرا می‌شود؛ زیرا مرحله ترجمه فقط یک مرتبه انجام می‌شود.
- قابلیت حمل: از آنجایی که یک برنامه کامپایل شده به کد ماشین ترجمه می‌شود، فقط می‌تواند بر روی نوع خاصی از سخت افزار یا سیستم عامل اجرا شود. در مقابل، یک برنامه تفسیر شده را می‌توان بر روی هر ماشینی اجرا کرد که یک مفسر برای زبان برنامه نویسی نصب شده باشد.
- تشخیص و گزارش خطا: کامپایلرها در طول فرایند کامپایل، خطاهای موجود در کد منبع را شناسایی کرده و به برنامه نویس گزارش می‌دهند. مفسرها خطاها را یک خط در طول فرایند اجرا شناسایی کرده و در صورت وقوع آنها را به برنامه‌نویس گزارش می‌دهند.
- استفاده از حافظه: برنامه‌های کامپایل شده معمولاً نسبت به برنامه های تفسیر شده به حافظه کمتری برا ی اجرا نیاز دارند؛ زیرا کل برنامه قبل از اجرا ترجمه می‌شود. از سوی دیگر، برنامه‌های تفسیر شده، به حافظه بیشتری نیاز دارند؛ زیرا مفسر باید خط جاری در حال اجرا و سایر اطلاعات وضعیت برنامه را پیگیری کند.
- اشکالزدایی(Debugging): اشکال‌زدایی برنامه‌های کامپایل شده می‌تواند دشوارتر از اشکال‌زدایی برنامه‌های تفسیر شده باشد؛ زیرا کد منبع در حین اجرا در دسترس نیست. از سوی دیگر، مفسرها می‌توانند پیام‌های خطای دقیقی را ارائه دهند و به برنامه‌نویس اجازه دهند تا در حین اجرا از کد عبور کند تا اشکال‌زدایی آسان‌تر انجام شود.
- انعطاف پذیری: مفسرها اغلب نسبت به کامپایلرها انعطاف پذیرتر هستند، زیرا یک فایل اجرایی باینری مجزا تولید نمی کنند، می‌توانند کد را به صورت پویا اجرا کنند و امکان توسعه تعاملی را فراهم کنند و سبک های برنامه‌نویسی پویاتری را ایجاد کنند.

۳

الف

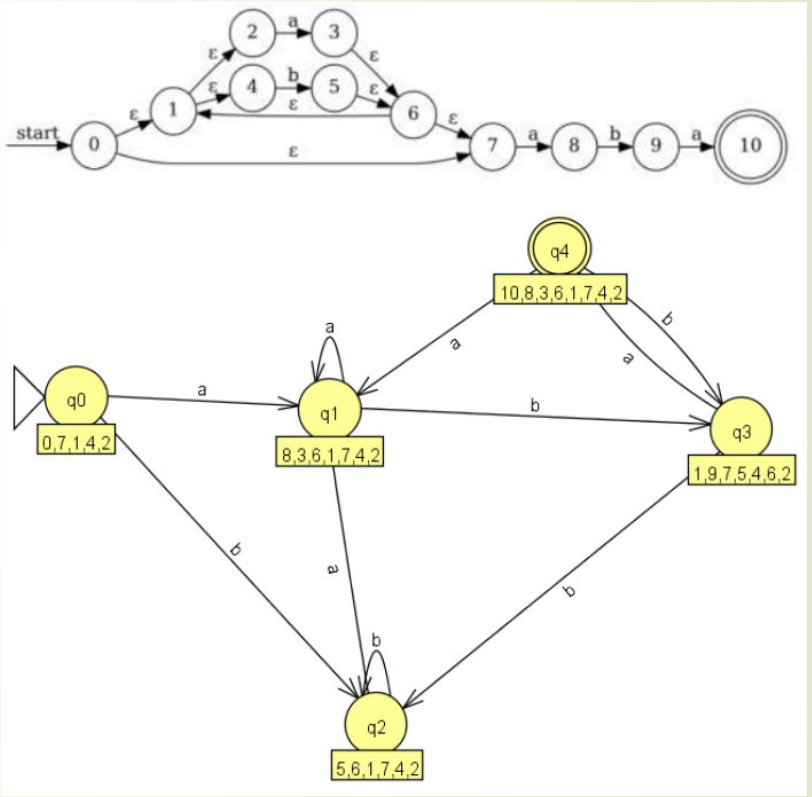
- $(\Sigma^*1)^*(00)^+((1\Sigma^*1)|1)(00)^+(1\Sigma^*)^*$

ب



۴-الف

- $\varepsilon - Closure\{0\} = \{0,1,2,4,7\}$
- $\varepsilon - Closure\{1\} = \{1,2,4\}$
- $\varepsilon - Closure\{2\} = \{2\}$
- $\varepsilon - Closure\{3\} = \{1,2,3,4,6,7\}$
- $\varepsilon - Closure\{4\} = \{4\}$
- $\varepsilon - Closure\{5\} = \{1,2,4,5,6,7\}$
- $\varepsilon - Closure\{6\} = \{1,2,4,6,7\}$
- $\varepsilon - Closure\{7\} = \{7\}$
- $\varepsilon - Closure\{8\} = \{8\}$
- $\varepsilon - Closure\{9\} = \{9\}$
- $\varepsilon - Closure\{10\} = \{10\}$



۴–ب

گام اول:

State	a	b
s	{s,q,p}	f
f	q	f
p	f	{p,q}
q	f	{p,q}

گام دوم:

δ({s,q,p},a)= δ(s,a)Uδ(q,a)Uδ(p,a)={s,q,p}U{f}U{f}={s,q,p,f} استیت جدید
δ({s,q,p},b)= δ(s,b)Uδ(q,b)Uδ(p,b)={f}U{p,q}U{p,q}={f,q,p} استیت جدید

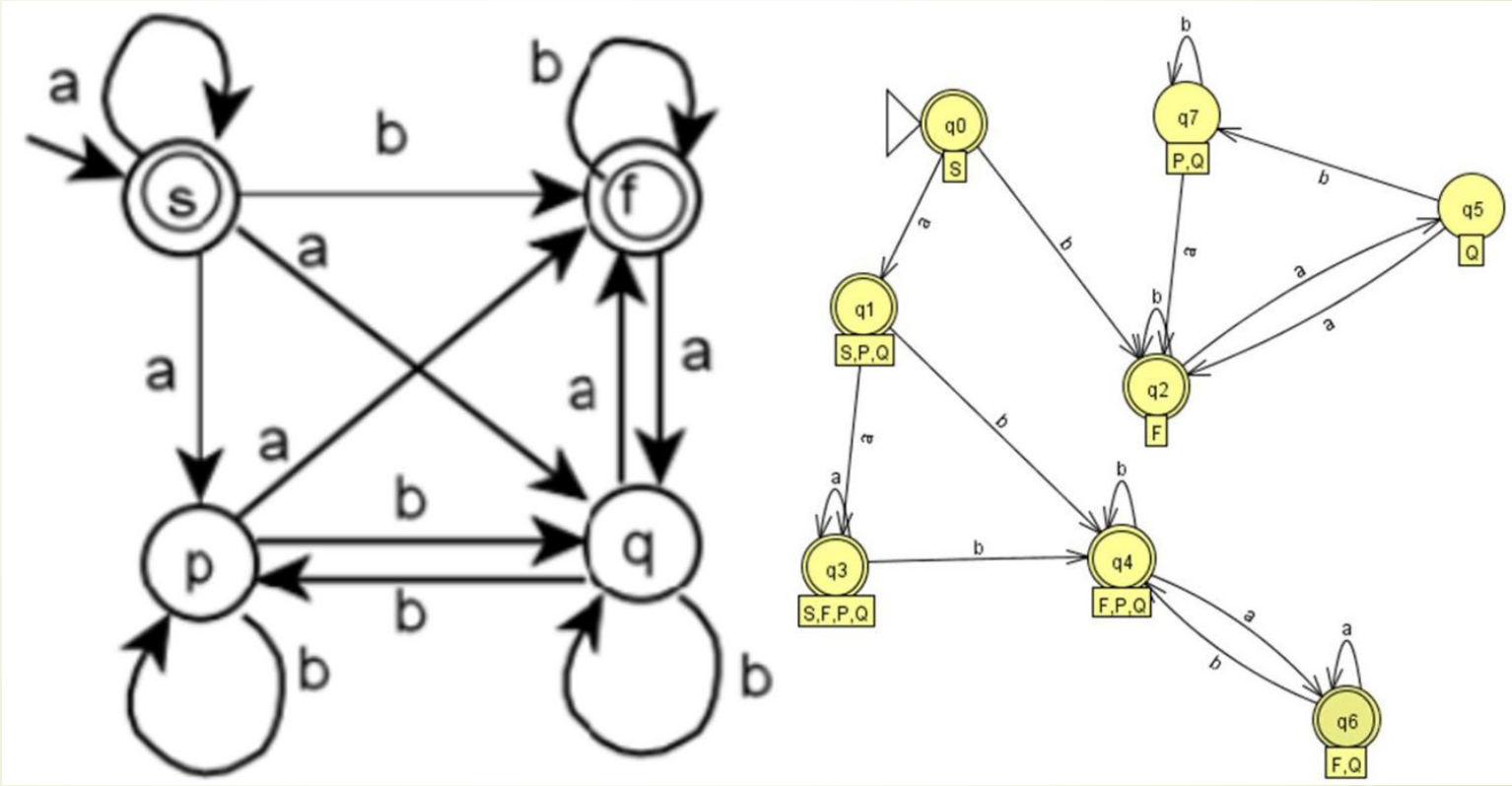
δ({p,q},a)= δ(q,a)Uδ(p,a) ={f}U{f}={f}
δ({p,q},b)= δ(q,b)Uδ(p,b)={p,q}U{p,q}={p,q}

δ({f,q,p},a)=δ(f,a)Uδ(q,a)Uδ(p,a)={q}U{f)U{f}={q,f} استیت جدید
δ({f,q,p},b) =δ(f,b)Uδ(q,b)Uδ(p,b)={f}U{p,q}U{p,q}={f,q,p}

δ({s,q,p,f},a)=δ(s,a)Uδ(f,a)Uδ(q,a)Uδ(p,a)={s,q,p}U{q}U{f)U{f}={s,q,p,f}
δ({s,q,p,f},b) =δ(s,a)Uδ(f,b)Uδ(q,b)Uδ(p,b)={f}U{f}U{p,q}U{p,q}={f,q,p}

δ({q,f},a)=δ(q,a)Uδ(f,a)={q}U{f}={q,f}
δ({q,f},b)=δ(q,b)Uδ(f,b)={p,q}U{f}={f,q,p}

۴–ب (ادامه)



۵-الف

گام اول:

- All states:

	a	b
q0	q3	q1
q1	q2	q3
q2	q4	q4
q3	q3	q1
q4	q3	q5
q5	q0	q4

گام دوم:

- Final states:

	a	b
q0	q3	q1
q2	q4	q4
q3	q3	q1

- Non-Final states:

	a	b
q1	q2	q3
q4	q3	q5
q5	q0	q4

۵-الف (ادامه)

گام سوم:

- Final State:

	a	b
q0,q3	q0,q3	q1
q2	q4	q4

- Non-Final State:

	a	b
q1	q2	q0,q3
q4	q0,q3	q5
q5	q0,q3	q4

گام چهارم:

- Final State:

	a	b
q0,q3	q0,q3	q1
q2	q4	q4

- Non-Final State:

	a	b
q1	q2	q0,q3
q4,q5	q0,q3	q4,q5

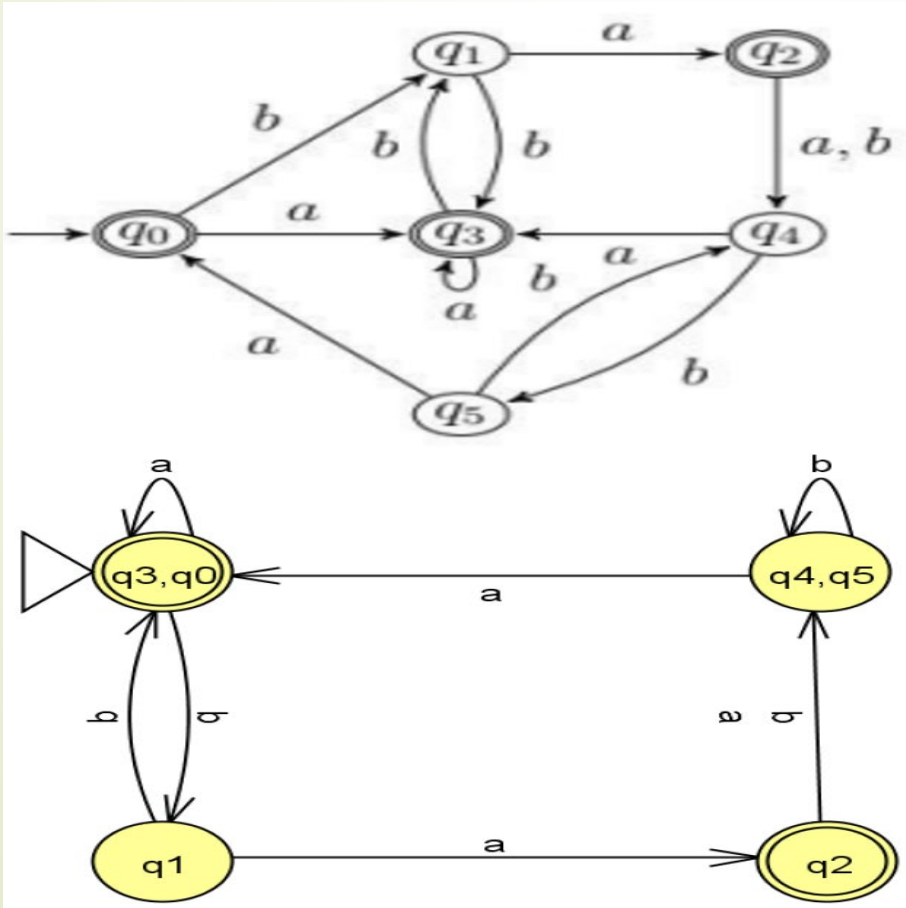
۵-الف (ادامه)

گام پنجم:

- ادغام دو جدول:

	a	b
q0,q3	q0,q3	q1
q2	q4	q4
q1	q2	q0,q3
q4,q5	q0,q3	q4,q5

شکل:



۵-الف (ادامه)

level 1: $G_1 = \{q_1, q_4, q_5\}$, $G_2 = \{q_0, q_2, q_3\}$
level 2(G_1): $\sigma(q_1, a) = q_2 \in G_2, \sigma(q_4, a) = q_3 \in G_2$ ok
$\sigma(q_1, b) = q_3 \in G_2, \sigma(q_4, b) = q_5 \in G_1$ so we should separate q_1 and q_4.

$\sigma(q_1, a) = q_2 \in G_2, \sigma(q_5, a) = q_0 \in G_2$ $\sigma(q_1, b) = q_3 \in G_2, \sigma(q_5, b) = q_4 \in G_1$ so we should separate q_1 and q_5.

$\sigma(q_5, a) = q_0 \in G_2, \sigma(q_4, a) = q_3 \in G_2$ $\sigma(q_5, b) = q_4 \in G_1, \sigma(q_4, b) = q_5 \in G_1$ so q_4 and q_5 are indistinguishable and stick together.

level 2(G_2): $\sigma(q_0, a) = q_3 \in G_2, \sigma(q_2, a) = q_4 \in G_1$ so we should separate q_0 and q_2.
$\sigma(q_0, a) = q_3 \in G_2, \sigma(q_3, a) = q_3 \in G_2$
$\sigma(q_0, b) = q_1 \in G_1, \sigma(q_3, b) = q_1 \in G_1$ so q_3 sticks with q_0.

level 3: $G_{11} = \{q_1\}, G_{12} = \{q_4, q_5\}, G_{21} = \{q_2\}, G_{22} = \{q_0, q_3\}$
$\sigma(q_4, a) = q_3 \in G_{22}, \sigma(q_5, a) = q_0 \in G_{22}$ ok
$\sigma(q_4, b) = q_5 \in G_{12}, \sigma(q_5, b) = q_4 \in G_{12}$ no separation here

$\sigma(q_0, a) = q_3 \in G_{22}, \sigma(q_3, a) = q_3 \in G_{22}$ok
$\sigma(q_0, b) = q_1 \in G_{11}, \sigma(q_3, b) = q_1 \in G_{11}$. no separation.

The algorithm converges and we have 4 states

۵–ب

گام اول:

- All states:

	a	B
q1	q2	q3
q2	q4	q5
q3	q6	q7
q4	q5	q4
q5	q7	q5
q6	q2	q7
q7	q7	q4

گام دوم:

- Final states:

	a	b
q1	q2	q3
q2	q4	q5
q7	q7	q4

- Non-Final states:

	a	b
q3	q6	q7
q4	q5	q4
q5	q7	q5
q6	q2	q7

۵–ب (ادامه)

- {q1,q7}, {q2}, {q3}, {q4}, {q5}, {q6}, {q7}

	a	b
q1	q2	q3
q7	q7	q4

- هیچ دو حالت قابل ادغامی یافت نمی‌شود، بنابراین ماشین متناهی قطعی در حالت کمینه می‌باشد.

۵–ب (ادامه)

level 1: $G_1 = \{q_1, q_2, q_7\}$, $G_2 = \{q_3, q_4, q_5, q_6\}$

level 2: $\sigma(q_1, a) = q_2 \in G_1, \sigma(q_2, a) = q_4 \in G_2$ separate q_1 and q_2.

$\sigma(q_1, a) = q_2 \in G_1, \sigma(q_7, a) = q_7 \in G_1$

$\sigma(q_1, b) = q_3 \in G_2, \sigma(q_7, b) = q_4 \in G_2$ q_1 and q_7 stay together.

$\sigma(q_3, a) = q_6 \in G_2, \sigma(q_4, a) = q_5 \in G_2$

$\sigma(q_3, b) = q_7 \in G_1, \sigma(q_4, b) = q_4 \in G_2$ separate q_3 and q_4

$\sigma(q_3, a) = q_6 \in G_2, \sigma(q_5, a) = q_7 \in G_1$separate q_3 and q_5

$\sigma(q_3, a) = q_6 \in G_2, \sigma(q_6, a) = q_2 \in G_1$separate q_3 and q_6

$\sigma(q_4, a) = q_5 \in G_2, \sigma(q_5, a) = q_7 \in G_1$separate q_4 and q_5

$\sigma(q_4, a) = q_5 \in G_2, \sigma(q_6, a) = q_2 \in G_1$ separate q_4 and q_6

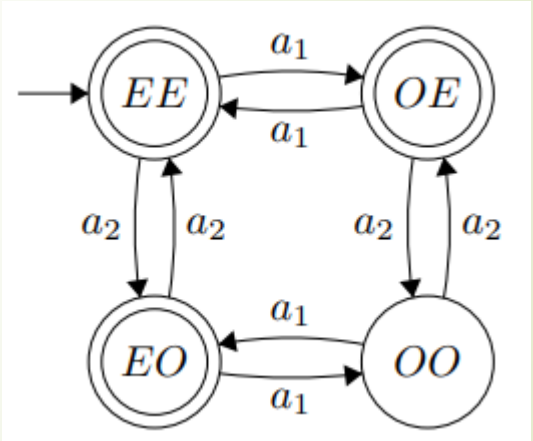
$\sigma(q_5, a) = q_7 \in G_1, \sigma(q_6, a) = q_2 \in G_1$ $\sigma(q_5, b) = q_5 \in G_2, \sigma(q_6, b) = q_7 \in G_1$ separate q_5 and q_6

level 3: $\{q_1, q_7\}\{q_2\}\{q_3\}\{q_4\}\{q_5\}\{q_6\}$

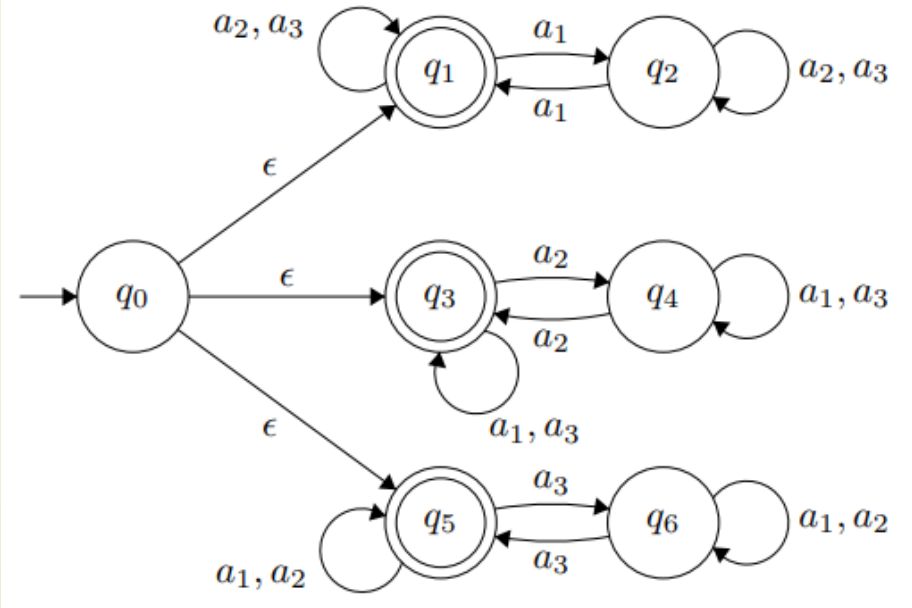
$\sigma(q_1, a) = q_2, \sigma(q_7, a) = q_7$ q_1 and q_7 are not in the same group, we separate them and then each group contains only one state. It means that the proposed DFA is of the minimum states and we can not minimize it anymore.

۶

الف


EE
OE
EO
OO
a_1
a_2

ب


q_0
q_1
q_2
q_3
q_4
q_5
q_6
ϵ
a_1
a_2
a_3
a_2, a_3
a_1, a_3
a_1, a_2

۷

الف

$(a|ba^*ba^*)^*$

ب

$aa^+bb^+|aaa^+b^+$

۷ (ادامه)

ج

$$(a(a|b)^{+}a)\,|\,(b(a|b)^{+}b)\,|$$

$$\begin{array}{ll}
(aa(a|b)^{+}aa)\,| & (ab(a|b)^{+}ab)\,| \\
(ba(a|b)^{+}ba)\,| & (bb(a|b)^{+}bb)\,| \\
(aaa(a|b)^{+}aaa)\,| & (aab(a|b)^{+}aab)\,| \\
(aba(a|b)^{+}aba)\,| & (abb(a|b)^{+}abb)\,| \\
(baa(a|b)^{+}baa)\,| & (bab(a|b)^{+}bab)\,| \\
(bba(a|b)^{+}bba)\,| & (bba(a|b)^{+}bba)\,| \\
(aaaa(a|b)^{+}aaaa)\,| & (aaab(a|b)^{+}aaab)\,| \\
(aaba(a|b)^{+}aaba)\,| & (aabb(a|b)^{+}aabb)\,| \\
(abaa(a|b)^{+}abaa)\,| & (abab(a|b)^{+}abab)\,| \\
(abba(a|b)^{+}abba)\,| & (abbb(a|b)^{+}abbb)\,| \\
(baaa(a|b)^{+}baaa)\,| & (baab(a|b)^{+}baab)\,| \\
(baba(a|b)^{+}baba)\,| & (babb(a|b)^{+}babb)\,| \\
(bbaa(a|b)^{+}bbaa)\,| & (bbab(a|b)^{+}bbab)\,| \\
(bbba(a|b)^{+}bbba)\,| & (bbbb(a|b)^{+}bbbb)
\end{array}$$

د

$$(aa)^{*}(a|b)(bb)^{*}$$

۸–الف

- تعداد: ۳۵ + EOF = ۳۶

```
main ( )
{
   int * a , b ;
   b = 10 ;
   a = & b ;
   printf ( "%d%d" , b , * a ) ;
   b = * /*pointer*/ b ;
}
```

- انواع

ID - LPAREN – RPAREN - LBRACE – INT – STAR - ID - COMMA - ID - SEMI - ID - ASSIGN – NUM – SEMI - ID - ASSIGN - AND – ID – SEMI - ID - LPAREN – STRING - COMMA – ID – COMMA – STAR – ID - RPAREN - SEMI - ID - ASSIGN – STAR - ID - SEMI - RBRACE - EOF

۸–ب

- تعداد:۳۳+ EOF = ۳۴

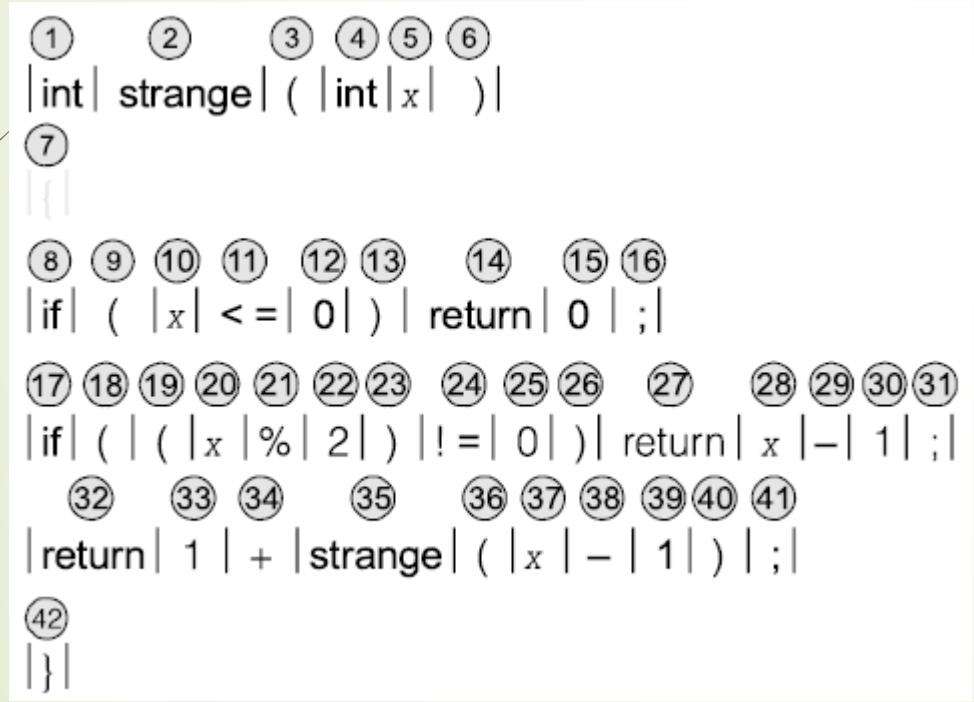
```
main ( )
①  ②③
{
④
    char ch = 'A' ;
    ⑤  ⑥⑦ ⑧⑨
    int x , y ;
    ⑩ ⑪ ⑫⑬ ⑭
    x = y = 20 ;
    ⑮⑯ ⑰ ⑱ ⑲ ⑳
    x ++ ;
    ㉑ ㉒ ㉓
    printf ( "%d%d" , x , y ) ;
    ㉔  ㉕  ㉖    ㉗㉘ ㉙ ㉚㉛ ㉜
}
㉝
```

- انواع:

ID - LPAREN - RPAREN - LBRACE - CHAR - ID - ASSIGN - NUM - SEMI - INT - ID - COMMA - ID - SEMI - ID - ASSIGN - ID - ASSIGN - NUM - SEMI - STRING - COMMA - ID - COMMA - ID - ID - INC - SEMI - ID - LPAREN - RPAREN - SEMI - RBRACE - EOF

۸–ج

- تعداد: ۴۲ + EOF = ۴۳



- انواع:

INT – ID(strange) – LPAREN – INT – ID(x) – RPAREN – LBRACE – IF - LPAREN - ID(x) - LEQ - NUM(0) - RPAREN - RETURN - NUM(0) - SEMI - IF - LPAREN - LPAREN - ID(x) - MOD - NUM(2) – RPAREN - NOTEQ - NUM(0) - RPAREN - RETURN - ID(x) - MINUS - NUM(1) - SEMI - RETURN - NUM(1) - PLUS - ID(strange) - LPAREN - ID(x) - MINUS - NUM(1) - RPAREN - SEMI - RBRACE - EOF