دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

درس کامپایلر

تکلیف عملی سوم

تاریخ تحویل:۹ تیر ۱۴۰۲

سوال ۱.در این تمرین قصد داریم به کمک ابزار flex و bison یک ماشین حساب ساده برای اعداد اعشاری بسازیم که توانایی انجام عملیات های جمع ، تفریق ، ضرب ، تقسیم و توان را داشته باشد.

```
<expr> ::= <expr> '+' <expr>
        | <expr> '-' <expr>
        | <expr> '*' <expr>
        | <expr> '/' <expr>
        | <expr> '^' <expr>
        | '-' <expr>
        | '(' <expr> ')'
        | IDENTIFIER
        | NUMBER
```

برای ساخت این ماشین حساب میتوانید از گرامر بالا استفاده کنید و موارد دیگر را به گرامر اضافه کنید.

- توجه کنید گرامر مبهم است.
- در نام گذاری متغیرها از حروف و اعداد میتوان استفاده کرد.
- درصورتی که در ابتدای نام گذاری یک متغیر از عدد استفاده شود خطای مناسب و شماره خط خطا چاپ شود.
- ماشین حساب شما باید توانایی ذخیره ی اعداد در متغیرها و انجام عملیات های محاسباتی با متغیرها را داشته باشد.
- درصورت تقسیم یک عدد بر صفر، خطای مناسب و شماره خط خطا چاپ شود.
- به اولویت عملگرها دقت کنید.
- عملیات ها میتوانند شامل پرانتز برای افزایش اولویت محاسبات باشند.
- در صورت استفاده از متغیري که قبلا مقداردهی نشده ، پیغام خطای مناسب به همراه شماره خط خطا چاپ شود.
- در صورت استفاده از کاراکترهای خارج از زبان گرامر، پیغام خطای غیر معتبر بودن کاراکتر به همراه شماره خط خطا چاپ شود.

ورودی برنامه:یک فایل با پسوندtxt. است که در هر خط آن تعدادی عملیات محاسباتی نوشته شده است.

خروجی: برنامه ي شما باید براي هر خط از فایل ورودی که شامل کلمه کلیدی print است، نتیجه نهایي عملیات نوشته شده در print را در ترمینال چاپ کند.

اختیاری: ماشین حساب شما، توانایی انجام عملیات های زیر را داشته باشد.

- sin(x) , cos(x) ,tan(x) ,cot(x), log(x), exp(x)

مثال:

```
input.txt:
x = 10 / 2
y = x * 4
print(y - 6)
print(x + (-2 * 3))
a32 = -2 * 5 + 2^3
print(a32)
print((x-3)^2^3)
z = 0
print(x/z*2)
print(-2^3)
```

```
output:
14.000000
-1.000000
-2.000000
256.000000
Error at line 9: Division by zero
10.000000
-8.000000
```

input2.txt:

```
x = sin(2.5)
y = cos(2*x + 10.2)
print(x)
print(y)
print(tan(x))
print(cot(y))
print(log(x) * exp(x))
print(exp(x))
```

output:

```
0.598472
0.390680
0.681896
2.428069
-0.934003
1.819337
```

input3.txt:

```
x=2
y= x*2
z$ = -3.2
print(x+y)
print(x-5*3)
print(y-2^3)
```

output:

```
Error at line 3: invalid character: $
6.000000
-13.000000
-4.000000
```

input4.txt:

```
a = (10 / 2)
b = a * -4
print(a - b)
print(c^a)
c=2.5 * 4
print(c^a)
4wrong_var = 32/8
```

output:

```
25.000000
Error at line 4: variable 'c' has not been assigned before
100000.00000
Error at line 7 : invalid variable name
```

سوال ۲. در این تمرین قصد داریم به کمک ابزار flex و bison برنامه ای بنویسیم که توانایی پارس کردن گرامر داده شده را داشته باشد. این گرامر نشان دهنده ی یک زبان برنامه نویسی ساده است که قابلیت های زیر را دارد.

۱.نوع داده:در این گرامر از دونوع داده Boolean و Integer میتوانیم استفاده کنیم که با کلمات کلیدی int و bool مشخص میشوند.

۲.تعریف متغیرها: به کمک این گرامر میتوانیم متغیرها را declare کنیم.

مثال:

```
int x;
int z,y;
bool flag, isCorrect2;
```

- امکان تعریف چندین متغیر بانوع یکسان به صورت پشت سرهم که با کاما جداشدند، وجود دارد.
- برای نام گذاری متغیرها از قواعد موجود در گرامر استفاده کنید.

۳.مقداردهی متغیرها: به کمک این گرامر میتوانیم به متغیرها مقادیر مناسب اختصاص دهیم.

مثال:

```
int x;
x = 2;
bool flag;
flag = true;
```

- طبق گرامر امکان مقداردهی چندین متغیر پشت سرهم وجود ندارد و هربار فقط یک متغیر مقداردهی میشود.
- طبق گرامر، امکان مقداردهی، هم زمان با تعریف متغیر وجود دارد مانند مثال زیر:

```
bool z = (x + y) < (-2 * x);
```

۴.عبارت های شرطی:

در این گرامر میتوانیم از if و else به شکل زیر استفاده کنیم:

```
if(expression){
        //some code
}else{
        //some code
}
```

مثال:

```
if ((a > b) && (a <= c))
{
    x = 3;
    counter = 0;
}
else
{
    isCorrect = !flag;
}
```

- وجود else اختیاری است و ممکن است یک if بدون else ظاهر شود.
- در بدنه ی if و else میتواند تعداد صفر یا بیشتر statement قرار بگیرد.

۵. حلقه ها:

در این گرامر حلقه ها به شکل زیر تعریف میشوند:

```
while(expression){
        //some code
}
```

- شرطی که برای حلقه while بررسی میشود همانند شرطی است که برای if بررسی میشود.
- بدنه ی while هم مانند بدنه ی if و else است یعنی میتواند شامل صفر یا چندین statement باشد.

۶.عملگرها:

- Arithmetic Operators: شامل عملگرهای +, - , * , /
- Relational Operators: شامل عملگرهای ==, != , > , < , >= , <=
- Logical Operators: شامل عملگرهای ! , || , &&

۷.نکات تکمیلی:

- "foo": مقادیری که در دابل کوتیشن قرار گرفته اند، نشان دهنده ی ترمینال ها هستند.
- <foo>: مقادیری که داخل <> قرار دارند نشان دهنده ی non-terminal ها هستند.
- *<foo>: مقادیری که با * نشان داده شدند به معنای ظاهرشدن صفر یا تعداد بیشتری از foo است.
- +<foo>: مقادیری که با + نشان داده شدند به معنای ظاهرشدن یک یا تعداد بیشتری از foo است.
- در مواردی که گرامر مبهم است، رفع ابهام لازم انجام شود.
- برنامه نهایی نباید دارای shift/reduce conflict یا reduce/reduce conflict باشد.

گرامر

<program> ::= <statement>*

<statement> ::= <declaration>

| <assignment>

| <conditional>

| <loop>

<type> ::= "int" | "bool"

<declaration> ::= <type> <identifier>+ "," ";"

<assignment> ::= <identifier> "=" <expression> ";"

| <type> <identifier> "=" <expression> ";"

<conditional> ::= "if" "(" <expression> ")" "{" <statement>* "}"

| "if" "(" <expression> ")" "{" <statement>* "}" "else" "{" <statement>* "}"

<loop> ::= "while" "(" <expression> ")" "{" <statement>* "}"

<expression> ::= <identifier>

| <literal>

| <expression> <binary_operator> <expression>

| "!" <expression>

| "(" <expression> ")"

| "-" <expression>

<literal> ::= <boolean_literal> | <integer_literal>

<boolean_literal> ::= "true" | "false"

<integer_literal> ::= <digit>+

<binary_operator> ::= <arithmetic_operator> | <relational_operator> | <logical_operator>

<arithmetic_operator> ::= "+" | "-" | "*" | "/"

<relational_operator> ::= "<" | ">" | "<=" | ">=" | "==" | "!="

<logical_operator> ::= ”||” | “&&”

<identifier> ::= <letter> <letter_or_digit>*

<letter_or_digit> ::= <letter> | <digit>

<digit> ::= "0" | "1" | ... | "9"

<letter> ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

جدول توکن های گرامر

Token	Token name
int	INT
bool	BOOL
if	IF
else	ELSE
while	WHILE
true	TRUE
false	FALSE
||	OR
&&	AND
=	ASSIGN
==	EQ
!=	NEQ
<=	LTE
>=	GTE
<	LT
>	GT
+	PLUS
-	MINUS
*	TIMES
/	DIVIDE
!	NOT
(	LPAREN
)	RPAREN
{	LBRACE
}	RBRACE
;	SEMICOLON

,	COMMA
[0-9]+	INTEGER_LITERAL
characters [characters ,integers]* example: a123	IDENTIFIER

۸.ورودی برنامه:

- ورودی به صورت یک فایل با پسوند txt. است که حاوی کدی است که با این زبان نوشته شده است.
- چنانچه یک برنامه ی صحیح به تحلیلگر شما داده شود باید بتواند Syntax Tree آن را استخراج کند و به صورت پیشوندی (pre-order) چاپ کند.
- به تحلیلگر شما یک آرگومان پاس داده میشود که اگر مقدار این آرگومان 1باشد ، باید Token Nameبرای ترمینال ها نمایش داده شود و در صورتی که این آرگومان 0باشد ، باید Token Valueبرای ترمینال ها چاپ شود.
- بدین ترتیب ، تحلیلگر شما باید نام فایل و این آرگومان را به صورت زیر دریافت کند.

$ ./syntaxParser input.txt 0 or 1

مثال ورودی:

```
int x, y;
x = 5;
y = x * 2 + 5;
bool isGreater;
isGreater = y > x;
if ((x != y) || (x > -2 && y < 30))
{
    int equation1;
    equation1 = 3 * x + y;
}
else
{
    int equation2;
    equation2 = -2 * x - 5;
}
```

۹.خروجی:

درصورتی که ورودی از گرامر زبان پیروی میکند، خروجی مناسب طبق ارگومان ورودی چاپ شود.

درصورت وجود خطا در ورودی، شماره ی خط خطا و پیام مناسب برای نشان دادن آن خطا چاپ شود.

خروجی برای ارگومان 1(چاپ Syntax Tree با نام توکن ها):

```
<program> <statement_list> <statement> <declaration> <type> INT <identifier_list> <identifier_list> <identifier> IDENTIFIER COMMA
<identifier> IDENTIFIER SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> INTEGER_LITERAL
SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> <expression> <expression> <identifier>
IDENTIFIER TIMES <expression> INTEGER_LITERAL PLUS <expression> INTEGER_LITERAL SEMICOLON <statement_list> <statement> <declaration> <type>
BOOL <identifier_list> <identifier> IDENTIFIER SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN
<expression> <expression> <identifier> IDENTIFIER GT <expression> <identifier> IDENTIFIER SEMICOLON <statement_list> <statement>
<conditional> IF LPAREN <expression> <expression> LPAREN <expression> <expression> <identifier> IDENTIFIER NEQ <expression> <identifier>
IDENTIFIER RPAREN OR <expression> LPAREN <expression> <expression> <expression> <identifier> IDENTIFIER GT <expression> <expression> MINUS
<expression> INTEGER_LITERAL AND <expression> <identifier> IDENTIFIER LT <expression> INTEGER_LITERAL RPAREN RPAREN LBRACE <statement_list>
<statement> <declaration> <type> INT <identifier_list> <identifier> IDENTIFIER SEMICOLON <statement_list> <statement> <assignment>
<identifier> IDENTIFIER ASSIGN <expression> <expression> <expression> INTEGER_LITERAL TIMES <expression> <identifier> IDENTIFIER PLUS
<expression> <identifier> IDENTIFIER SEMICOLON <statement_list> RBRACE ELSE LBRACE <statement_list> <statement> <declaration> <type> INT
<identifier_list> <identifier> IDENTIFIER SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression>
<expression> <expression> MINUS <expression> INTEGER_LITERAL TIMES <expression> <identifier> IDENTIFIER MINUS <expression> INTEGER_LITERAL
SEMICOLON <statement_list> RBRACE <statement_list>
```

خروجی برای ارگومان 0(چاپ Syntax Tree با مقادیر توکن ها):

```
<program> <statement_list> <statement> <declaration> <type> int <identifier_list> <identifier_list> <identifier> x , <identifier> y ;
<statement_list> <statement> <assignment> <identifier> x = <expression> 5 ; <statement_list> <statement> <assignment> <identifier> y =
<expression> <expression> <expression> <identifier> x * <expression> 2 + <expression> 5 ; <statement_list> <statement> <declaration> <type>
bool <identifier_list> <identifier> isGreater ; <statement_list> <statement> <assignment> <identifier> isGreater = <expression>
<expression> <identifier> y > <expression> <identifier> x ; <statement_list> <statement> <conditional> if ( <expression> <expression> (
<expression> <expression> <identifier> x != <expression> <identifier> y ) || <expression> ( <expression> <expression> <expression>
<identifier> x > <expression> <expression> - <expression> 2 && <expression> <identifier> y < <expression> 30 ) ) { <statement_list>
<statement> <declaration> <type> int <identifier_list> <identifier> equation1 ; <statement_list> <statement> <assignment> <identifier>
equation1 = <expression> <expression> <expression> 3 * <expression> <identifier> x + <expression> <identifier> y ; <statement_list> } else
{ <statement_list> <statement> <declaration> <type> int <identifier_list> <identifier> equation2 ; <statement_list> <statement>
<assignment> <identifier> equation2 = <expression> <expression> <expression> - <expression> 2 * <expression> <identifier> x - <expression>
5 ; <statement_list> } <statement_list>
```

مثال۲:

ورودی:

```
int a;
bool flag, x, y;
x = true;
y = true;
flag = !(x && y) || y;
a = 2;
b = a > 2;
c = -2 * 3 == 2 * 3;
```

خروجی برای ارگومان 1(چاپ Syntax Tree با نام توکن ها):

```
<program> <statement_list> <statement> <declaration> <type> INT <identifier_list> <identifier> IDENTIFIER SEMICOLON <statement_list>
<statement> <declaration> <type> BOOL <identifier_list> <identifier_list> <identifier_list> <identifier> IDENTIFIER COMMA <identifier>
IDENTIFIER COMMA <identifier> IDENTIFIER SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression>
TRUE SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> TRUE SEMICOLON <statement_list>
<statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> <expression> NOT <expression> LPAREN <expression> <expression>
<identifier> IDENTIFIER AND <expression> <identifier> IDENTIFIER RPAREN OR <expression> <identifier> IDENTIFIER SEMICOLON <statement_list>
<statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> INTEGER_LITERAL SEMICOLON <statement_list> <statement> <assignment>
<identifier> IDENTIFIER ASSIGN <expression> <expression> <identifier> IDENTIFIER GT <expression> INTEGER_LITERAL SEMICOLON <statement_list>
<statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> <expression> <expression> MINUS <expression> INTEGER_LITERAL TIMES
<expression> INTEGER_LITERAL EQ <expression> <expression> INTEGER_LITERAL TIMES <expression> INTEGER_LITERAL SEMICOLON <statement_list>
```

خروجی برای ارگومان 0(چاپ Syntax Tree با مقادیر توکن ها):

```
<program> <statement_list> <statement> <declaration> <type> int <identifier_list> <identifier> a ; <statement_list> <statement>
<declaration> <type> bool <identifier_list> <identifier_list> <identifier_list> <identifier> flag , <identifier> x , <identifier> y ;
<statement_list> <statement> <assignment> <identifier> x = <expression> true ; <statement_list> <statement> <assignment> <identifier> y =
<expression> true ; <statement_list> <statement> <assignment> <identifier> flag = <expression> <expression> ! <expression> ( <expression>
<expression> <identifier> x && <expression> <identifier> y ) || <expression> <identifier> y ; <statement_list> <statement> <assignment>
<identifier> a = <expression> 2 ; <statement_list> <statement> <assignment> <identifier> b = <expression> <expression> <identifier> a >
<expression> 2 ; <statement_list> <statement> <assignment> <identifier> c = <expression> <expression> <expression> - <expression> 2 *
<expression> 3 == <expression> <expression> 2 * <expression> 3 ; <statement_list>
```

مثال۳:

ورودی:

```
bool flag;
flag = true;
int counter;
counter = 0;

while (flag)
{
    counter = counter + 1;
    if (counter == 10)
    {
        flag = !flag;
    }
}
```

خروجی برای ارگومان 1(چاپ Syntax Tree با نام توکن ها):

```
<program> <statement_list> <statement> <declaration> <type> BOOL <identifier_list> <identifier> IDENTIFIER SEMICOLON <statement_list>
<statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> TRUE SEMICOLON <statement_list> <statement> <declaration> <type> INT
<identifier_list> <identifier> IDENTIFIER SEMICOLON <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression>
INTEGER_LITERAL SEMICOLON <statement_list> <statement> <loop> WHILE LPAREN <expression> <identifier> IDENTIFIER RPAREN LBRACE
<statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN <expression> <expression> <identifier> IDENTIFIER PLUS
<expression> INTEGER_LITERAL SEMICOLON <statement_list> <statement> <conditional> IF LPAREN <expression> <expression> <identifier>
IDENTIFIER EQ <expression> INTEGER_LITERAL RPAREN LBRACE <statement_list> <statement> <assignment> <identifier> IDENTIFIER ASSIGN
<expression> NOT <expression> <identifier> IDENTIFIER SEMICOLON <statement_list> RBRACE <statement_list> RBRACE <statement_list>
```

خروجی برای ارگومان 0(چاپ Syntax Tree با مقادیر توکن ها):

```
<program> <statement_list> <statement> <declaration> <type> bool <identifier> flag ; <statement_list> <statement> <assignment> <identifier>
flag = <expression> true ; <statement_list> <statement> <declaration> <type> int <identifier> counter ; <statement_list> <statement>
<assignment> <identifier> counter = <expression> 0 ; <statement_list> <statement> <loop> while ( <expression> <identifier> flag ) {
<statement_list> <statement> <assignment> <identifier> counter = <expression> <expression> <identifier> counter + <expression> 1 ;
<statement_list> <statement> <conditional> if ( <expression> <expression> <identifier> counter == <expression> 10 ) { <statement_list>
<statement> <assignment> <identifier> flag = <expression> ! <expression> <identifier> flag ; <statement_list> } <statement_list> }
<statement_list>
```

مثال۴:

ورودی:

```
int x = 5;
int y = 7;
bool z = (x + y) < (-2 * x);
```

خروجی برای ارگومان 1(چاپ Syntax Tree با نام توکن ها):

```
<program> <statement_list> <statement> <assignment> <type> INT <identifier> IDENTIFIER ASSIGN <expression> INTEGER_LITERAL SEMICOLON
<statement_list> <statement> <assignment> <type> INT <identifier> IDENTIFIER ASSIGN <expression> INTEGER_LITERAL SEMICOLON <statement_list>
<statement> <assignment> <type> BOOL <identifier> IDENTIFIER ASSIGN <expression> <expression> LPAREN <expression> <expression> <identifier>
IDENTIFIER PLUS <expression> <identifier> IDENTIFIER RPAREN LT <expression> LPAREN <expression> <expression> MINUS <expression>
INTEGER_LITERAL TIMES <expression> <identifier> IDENTIFIER RPAREN SEMICOLON <statement_list>
```

خروجی برای ارگومان 0(چاپ Syntax Tree با مقادیر توکن ها):

```
<program> <statement_list> <statement> <assignment> <type> int <identifier> x = <expression> 5 ; <statement_list> <statement> <assignment>
<type> int <identifier> y = <expression> 7 ; <statement_list> <statement> <assignment> <type> bool <identifier> z = <expression>
<expression> ( <expression> <expression> <identifier> x + <expression> <identifier> y ) < <expression> ( <expression> <expression> -
<expression> 2 * <expression> <identifier> x ) ; <statement_list>
```

۱۰. اختیاری:

چاپ syntax tree به صورت دو بعدی(افقی یا عمودی) دارای نمره اضافه است.

چاپ درخت برای مثال ۴:

```
Abstract Syntax Tree:
<program>
   <statement_list>
      <statement>
         <assignment>
            <type>
               INT
            <identifier>
               IDENTIFIER
            ASSIGN
            <expression>
               INTEGER_LITERAL
            SEMICOLON
      <statement_list>
         <statement>
            <assignment>
               <type>
                  INT
               <identifier>
                  IDENTIFIER
               ASSIGN
               <expression>
                  INTEGER_LITERAL
               SEMICOLON
         <statement_list>
            <statement>
               <assignment>
                  <type>
                     BOOL
                  <identifier>
                     IDENTIFIER
                  ASSIGN
                  <expression>
                     <expression>
                        LPAREN

                        <expression>
                           <expression>
                              <identifier>
                                 IDENTIFIER
                           PLUS
                           <expression>
                              <identifier>
                                 IDENTIFIER
                        RPAREN
                     LT
                     <expression>
                        LPAREN
                        <expression>
                           <expression>
                              MINUS
                              <expression>
                                 INTEGER_LITERAL
                           TIMES
                           <expression>
                              <identifier>
                                 IDENTIFIER
                        RPAREN
                  SEMICOLON
            <statement_list>
```

```
Abstract Syntax Tree:
<program>
   <statement_list>
      <statement>
         <assignment>
            <type>
               int
            <identifier>
               x
            =
            <expression>
               5
            ;
      <statement_list>
         <statement>
            <assignment>
               <type>
                  int
               <identifier>
                  y
               =
               <expression>
                  7
               ;
         <statement_list>
            <statement>
               <assignment>
                  <type>
                     bool
                  <identifier>
                     z
                  =

                 <expression>
                    <expression>
                       (
                       <expression>
                          <expression>
                             <identifier>
                                x
                          +
                          <expression>
                             <identifier>
                                y
                       )
                    <
                    <expression>
                       (
                       <expression>
                          <expression>
                             -
                             <expression>
                                2
                          *
                          <expression>
                             <identifier>
                                x
                       )
                 ;
           <statement_list>
```

تذکرات نهایی:

- در فایل تکلیف، یک makefile قرار داده شده است.
- برای هردو سوال، متناسب با نام فایل های flex و bison خود، این فایل را اپدیت کنید.
- در انتها باید فایل های flex و bison و makefile را در پوشه ی مربوط به هر سوال قرارداده و کل تکلیف را به صورت فایل zip اپلود کنید.
- فیلم کوتاهی از توضیح کدهای خود واجرای آن ضبط کنید و آن را در پوشه همان سوال کنار فایل های دیگر قرار دهید.

نکات تکمیلی

۱. در این تکلیف استفاده از ابزار flex و bison الزامی است.

۲. فایل های مربوط به هر سوال را در پوشه ای با نام Qn قرار دهید که n شماره ی سوال مربوطه است.

۳. در هر پوشه باید فایل های flex و bison و makefile که متناسب با نام فایل های شما اپدیت شده به همراه فیلم توضیحات واجرای کد قرار داده شود.

۴.در انتها پوشه ی جواب ها را فشرده کنید و در قالب یک فایل zip اپلود کنید.

۵. فرمت نام گذاری تکلیف ارسالی باید به صورت زیر باشد :

HW3_Practical_LastName_StudentID که LastName نام خانوادگي شما و StudentID شماره دانشجويي شما است.

۶. انجام این تکلیف به صورت تك نفره است. در صورت مشاهده تقلب، نمرات هم مبدا کپی و هم مقصد آن صفر لحاظ می شود.

در صورت وجود هر گونه ابهام و یا سوال می توانید سوالات خود را در گروه تلگرام بپرسید. همچنین می توانید برای رفع ابهامات با دستیاران آموزشی در تماس باشید.

آیدی تلگرام:

- hadis_ghafouri
- alirezanum1

اسکایپ:

- live:.cid.f9217b1f0598c7b
- live:b1473ece83804d91