علیرضا اسلامی خواه 99521064

حالت اول برای این سوال انتخاب شده بود که در گام اول باید درخت را تکمیل تر میکردیم زیرا دو اشکال بزرگ داشت. اول اینکه برای switch مشخص نمیشد که روی چه متغیری انتخاب شده است و دوم اینکه برای case شرط را در درون درخت قرار نمیداد.

برای حل این مشکل درون گرامر دست به کار شدیم.

ابتدا سوییچ را عوض کردیم :

```
//switchst
//  returns[value_attr = str(), type_attr = str()]:
//  'switch' '(' expr ')' NEWLINE* '{' NEWLINE* case+ '}';

switchst
    returns[value_attr = str(), type_attr = str()]:
    'switch' '(' expr ')' NEWLINE* statement;
```

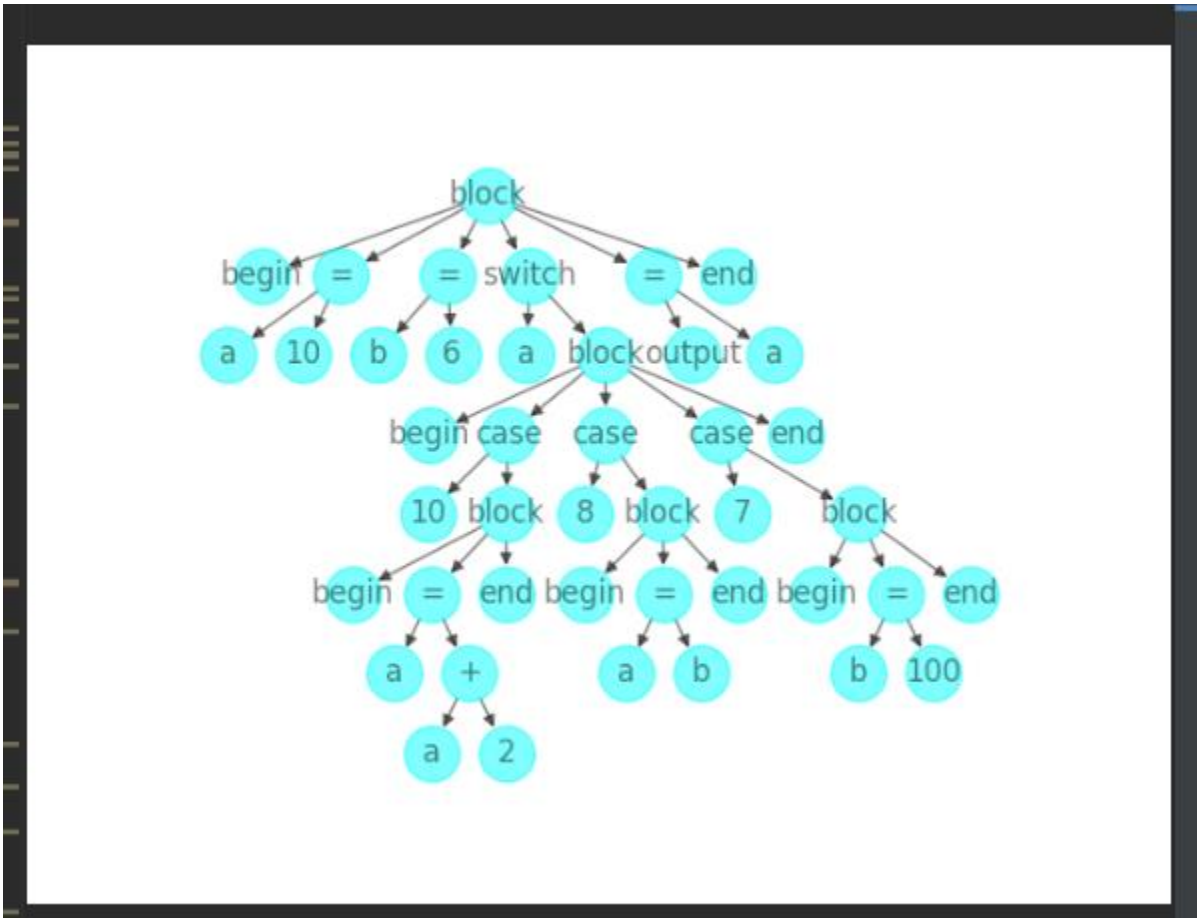
سپس درون statement کیس را هم قرار دادیم.

```
statement
    returns[value_attr = str(), type_attr = str()]:
    ifst
    | assign
    | compoundst
    | whilest
    | switchst
    | forst
    |case;
```

حال باید به جای INT نوعی expression میگذاشتیم تا value attrb جداگانه خودش را داشته باشد.

```
51
52  switchst
53      returns[value_attr = str(), type_attr = str()]:
54      'switch' '(' expr ')' NEWLINE* statement;
55
56  case
57      returns[value_attr = str(), type_attr = str()]:
58      'case' expr ':' statement NEWLINE*;
59
```

در مرحله بعد به سراغ کد il generator رفتیم که باز هم به چالش خوردیم و آن این بود که درون سوییچ یک بلاک کامل بود و نمیشد مبنایی برای مبدا آن در نظر گرفت.



در گرامر البته میتوانستیم break را هم جایگذاری کنیم به طوری که حتما در آخر case باشد اما کمبود وقت به ما اجازه نداد. ولی اینگونه میشد :

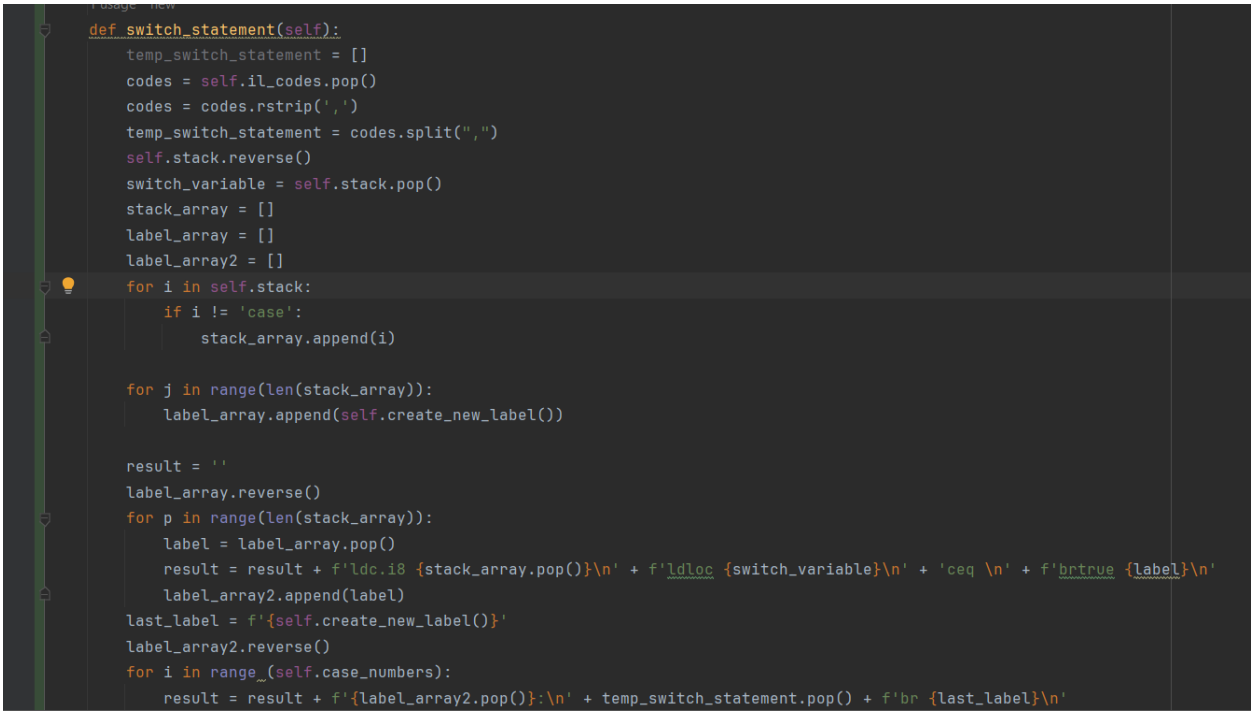
```
case
    returns[value_attr = str(), type_attr = str()]:
    'case' expr ':' statement NEWLINE* 'break';
```

ودرون switch statement هم بین گونه :

```
    label_array = []
    label_array2 = []
    for i in self.stack:
        if i != 'case' and i != 'break':
            stack_array.append(i)

    for j in range(len(stack_array)):
```

الگوریتم ما بدین صورت است که با توجه به درخت هرجا به case برخوردیم یک کاما(,) قرار دهیم. اینگونه میتوانیم کیس های یک بلاک را از هم تشخیص دهیم.



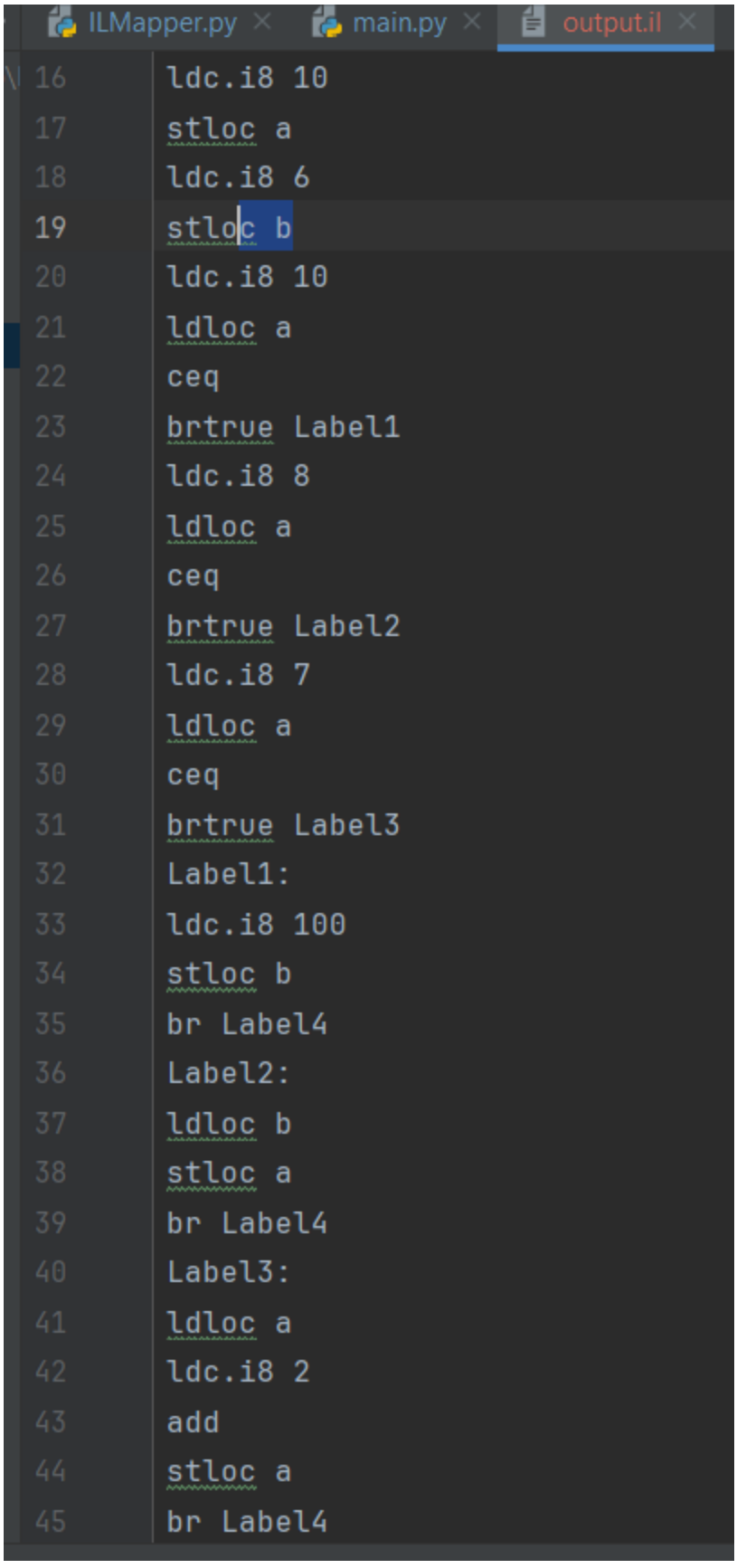
```
def switch_statement(self):
    temp_switch_statement = []
    codes = self.il_codes.pop()
    codes = codes.rstrip(',')
    temp_switch_statement = codes.split(",")
    self.stack.reverse()
    switch_variable = self.stack.pop()
    stack_array = []
    label_array = []
    label_array2 = []
    for i in self.stack:
        if i != 'case':
            stack_array.append(i)

    for j in range(len(stack_array)):
        label_array.append(self.create_new_label())

    result = ''
    label_array.reverse()
    for p in range(len(stack_array)):
        label = label_array.pop()
        result = result + f'ldc.i8 {stack_array.pop()}\n' + f'ldloc {switch_variable}\n' + 'ceq \n' + f'brtrue {label}\n'
        label_array2.append(label)
    last_label = f'{self.create_new_label()}'
    label_array2.reverse()
    for i in range(self.case_numbers):
        result = result + f'{label_array2.pop()}:\n' + temp_switch_statement.pop() + f'br {last_label}\n'
```

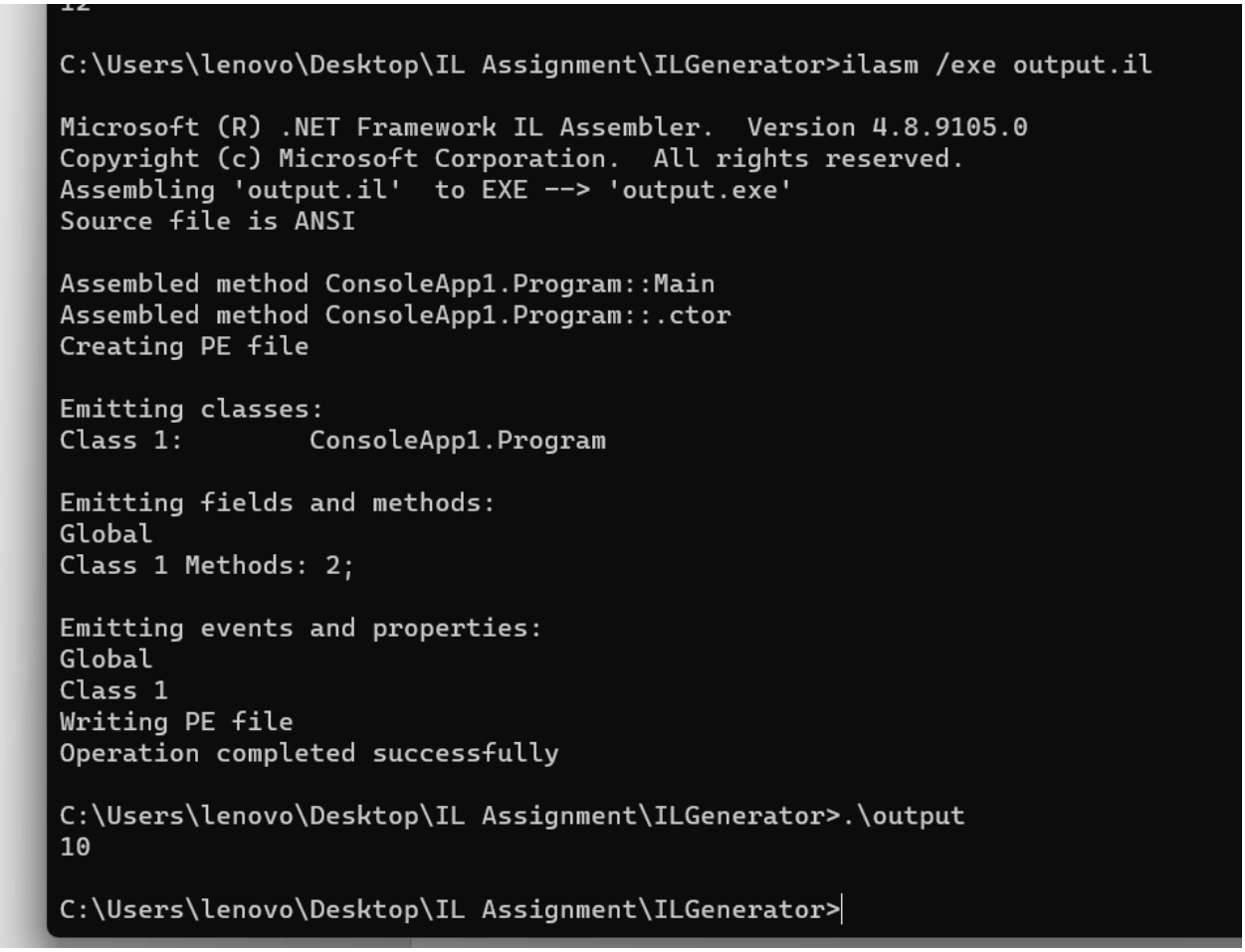
کد سوویچ کیس ما بدین صورت است که ابتدا با توجه به کاما هایی که گذاشتیم کیس هارا جدا میکنیم سپس متغیر سوییچ و شرط های آن را از استک جدا سازی میکنیم . **البته اینجا هم اگر breakداشتیم مانند بقیه case ها آن را از استک پاک میکردیم.**

در آخر هم کد را مانند چیزی که در ouput.il شرح داده شده تحویل میگیریم.


ILMapper.py
main.py
output.il
ldc.i8 10
stloc a
ldc.i8 6
stloc b
ldc.i8 10
ldloc a
ceq
brtrue Label1
ldc.i8 8
ldloc a
ceq
brtrue Label2
ldc.i8 7
ldloc a
ceq
brtrue Label3
Label1:
ldc.i8 100
stloc b
br Label4
Label2:
ldloc b
stloc a
br Label4
Label3:
ldloc a
ldc.i8 2
add
stloc a
br Label4

در اینجا هر شرط را بررسی کرده و با توجه به آن به برنچ مورد نظر پرش میکنیم.

در اخر هم نمایی از خروجی کامپایل شده کد il میبینیم:



ILMapper.py × main.py × output.il × in

```
1   program TestProgram
2   var
3       a : int
4       b : int
5       f : int
6   begin
7       a := 10
8       b := 6
9       switch (a)
10      begin
11          case 10 :begin
12              a := a + 2
13              end
14          case 8 :begin
15              a := b
16              end
17          case 7 : begin
18              b := 100
19              end
20      end
21      output := a
22
23  end
24
```