NAME : DIKSHA GUPTA

ROLL NO. : 27 | A2

PRACTICAL No. 5

Topic: Three Address Code Generation

Platform: Windows or Linux

Language to be used: Python or Java (based on the companies targeted for placement)

CO Mapped: CO4- Learn three address code generation and implement code optimization techniques for improving the performance of a program segment.

Aim: Write a program to generate three address code for the given language construct using SDTS.

a. Batch A1 : if-then-else,
b. Batch A2: for loop
c. Batch A3: while loop
d. Batch A4: do while loop

Input: Example for if-then-else

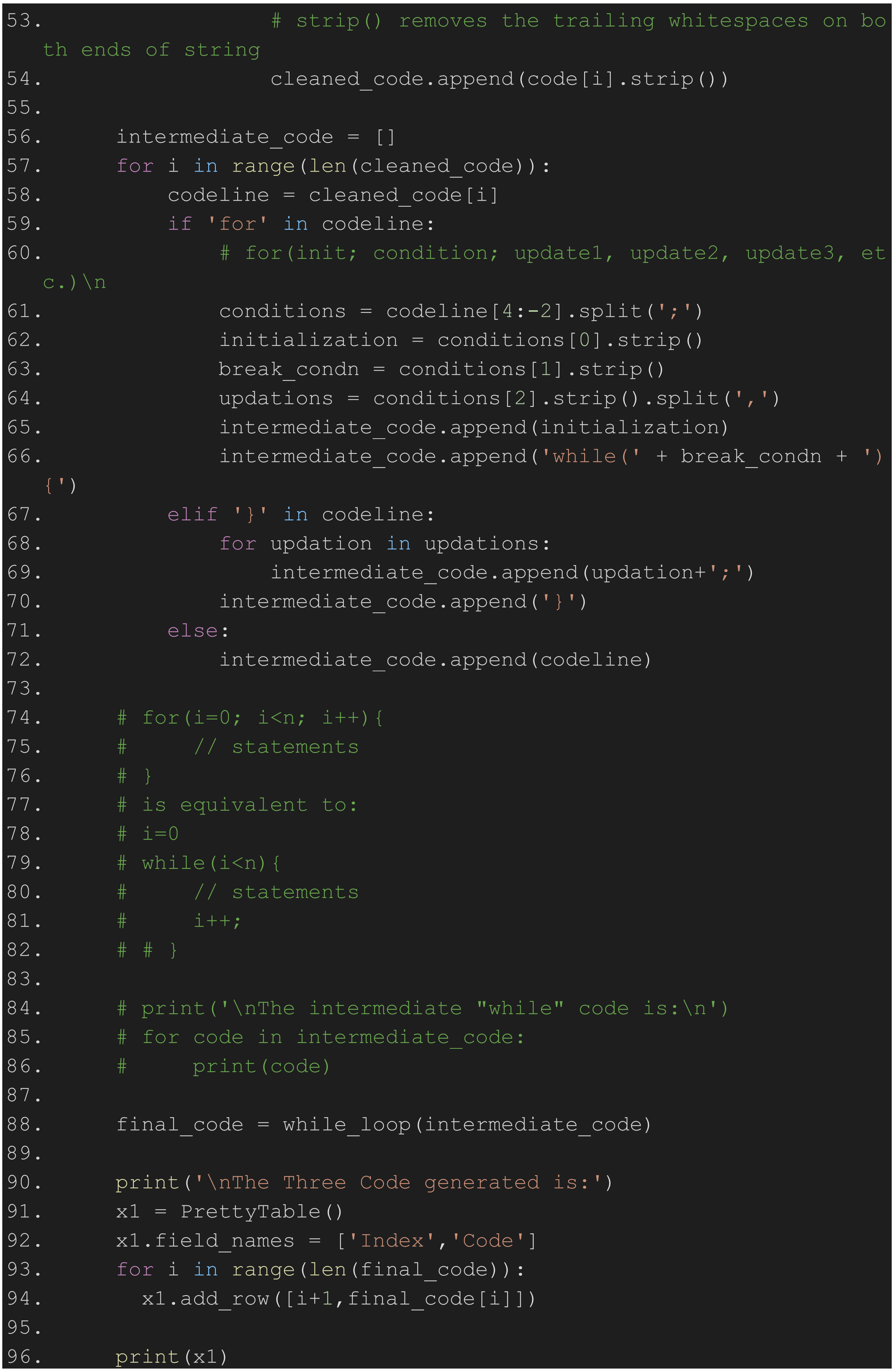
```
if (a<5)
{
        c= b+d
        d= i+j
}
else
{
         d= a+ b
        k= x+y

}
```

Output:

```
1.  if ( a<5) goto 3
2.  Goto_8
3.  T1=b+d
4.  c=T1
5.  T2=i+j
6.  d=T2
7.  goto 12
8.  T3=a+b
9.  d=T3
10.  T4=x+y
11.  k=T4
12. END
```

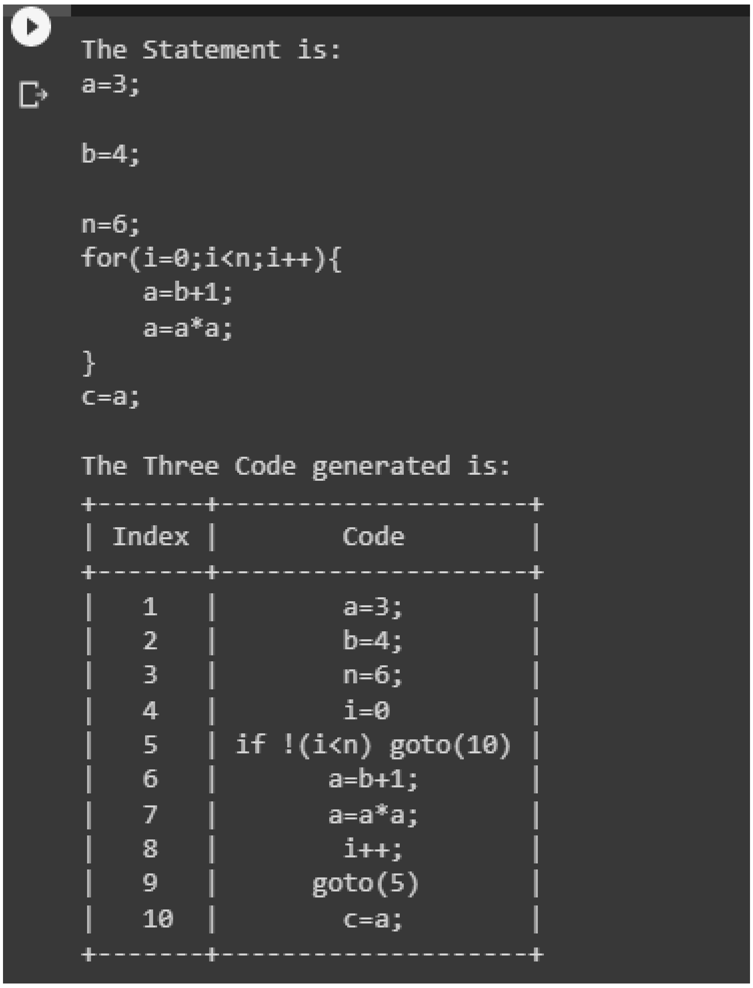
CODE :

```python
from prettytable import PrettyTable

def while_loop(cleaned_code):
    final_code = []
    while_idx = None
    for i in range(len(cleaned_code)):
        codeline = cleaned_code[i]

        if 'while' in codeline:
            while_idx = i
            # The loop condition would be enclosed in brackets
            start_idx = codeline.index('(')
            end_idx = codeline.index(')')
            # Select the substring between start_idx and end_idx
            bool_condn = ''.join(codeline[start_idx:end_idx+1])
            # Replace with
            final_code.append('if !{} goto({})'.format(bool_condn,None))
            while_idx = i
        elif '}' in codeline:
            final_code.append('goto({})'.format(while_idx+1))
            #
            final_code[while_idx] = final_code[while_idx].replace('None',str(i+2))
            while_idx = None
        else:
            final_code.append(codeline)
    return final_code

with open('code.txt') as f:
    code = f.readlines()

print('The Statement is:')
print(''.join(code))

cleaned_code = []
for i in range(len(code)):
    if code[i] != '\n':
        if code[i][-1] == '\n':
            # don't include the \n at the end of each line
            cleaned_code.append(code[i][:-1].strip())
        else:
```



```
                # strip() removes the trailing whitespaces on both ends of string
                cleaned_code.append(code[i].strip())

    intermediate_code = []
    for i in range(len(cleaned_code)):
        codeline = cleaned_code[i]
        if 'for' in codeline:
            # for(init; condition; update1, update2, update3, etc.)\n
            conditions = codeline[4:-2].split(';')
            initialization = conditions[0].strip()
            break_condn = conditions[1].strip()
            updations = conditions[2].strip().split(',')
            intermediate_code.append(initialization)
            intermediate_code.append('while(' + break_condn + '){')
        elif '}' in codeline:
            for updation in updations:
                intermediate_code.append(updation+';')
            intermediate_code.append('}')
        else:
            intermediate_code.append(codeline)

    # for(i=0; i<n; i++){
    #     // statements
    # }
    # is equivalent to:
    # i=0
    # while(i<n){
    #     // statements
    #     i++;
    # # }

    # print('\nThe intermediate "while" code is:\n')
    # for code in intermediate_code:
    #     print(code)

    final_code = while_loop(intermediate_code)

    print('\nThe Three Code generated is:')
    x1 = PrettyTable()
    x1.field_names = ['Index','Code']
    for i in range(len(final_code)):
      x1.add_row([i+1,final_code[i]])

    print(x1)
```

OUTPUT :



```
The Statement is:
a=3;

b=4;

n=6;
for(i=0;i<n;i++){
    a=b+1;
    a=a*a;
}
c=a;

The Three Code generated is:
+-------+--------------------+
| Index |        Code        |
+-------+--------------------+
|   1   |        a=3;        |
|   2   |        b=4;        |
|   3   |        n=6;        |
|   4   |        i=0         |
|   5   | if !(i<n) goto(10) |
|   6   |       a=b+1;       |
|   7   |       a=a*a;       |
|   8   |        i++;        |
|   9   |      goto(5)       |
|   10  |        c=a;        |
+-------+--------------------+
```