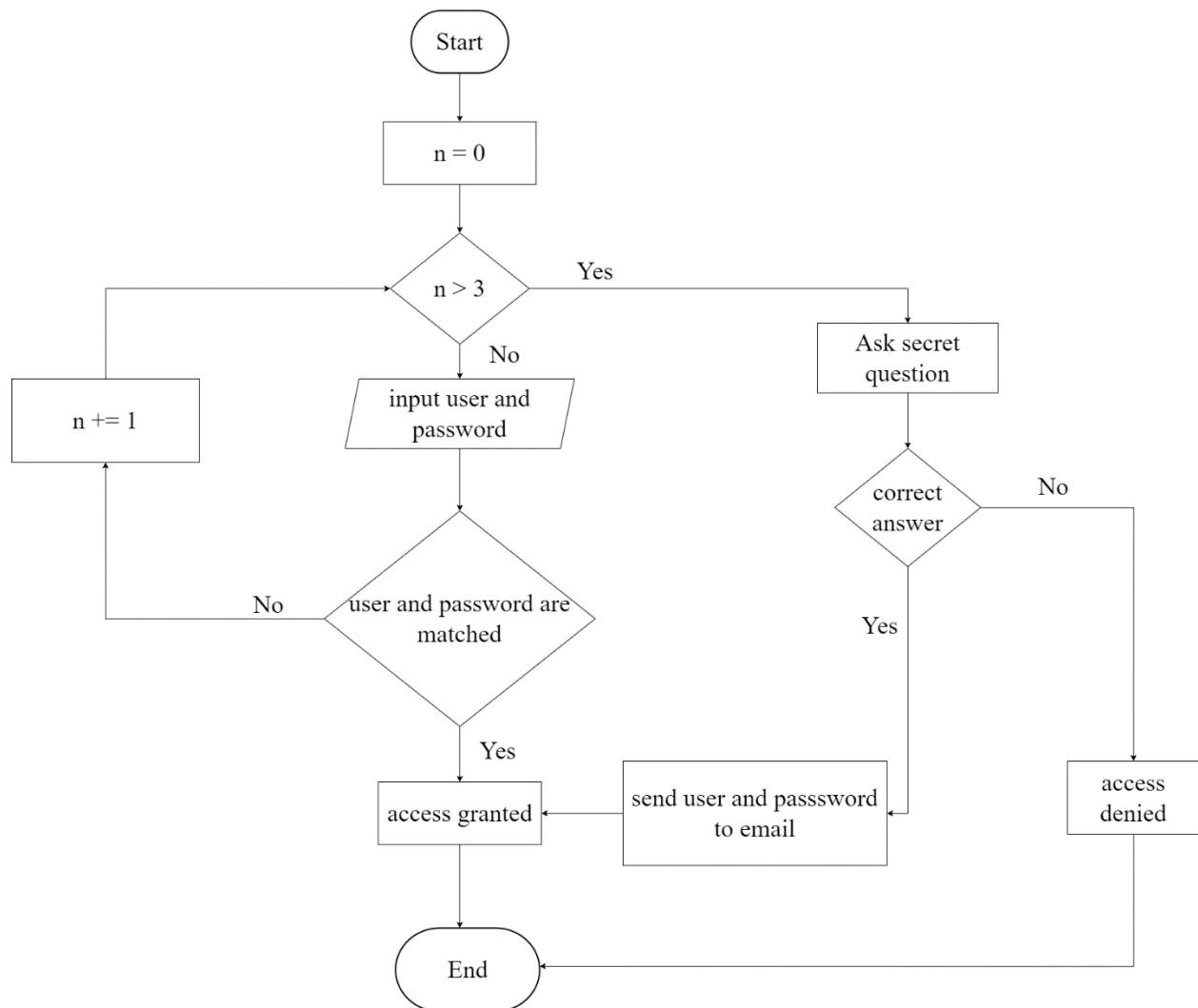


## Scenarios 1

- Flowchart



- Pseudocode

**SET** count(n) to 0

**WHILE** count(n) more than or equal to 3

**INPUT:** user enter input USER and PASSWORD

**IF** USER and PASSWORD are matched

access\_granted()

**ELSE**

Add 1 to count(n)

**ENDIF**

ANS = ask\_secret()

**IF** ANS == correct\_ans

access\_granted()

send\_email()

**ELSE**

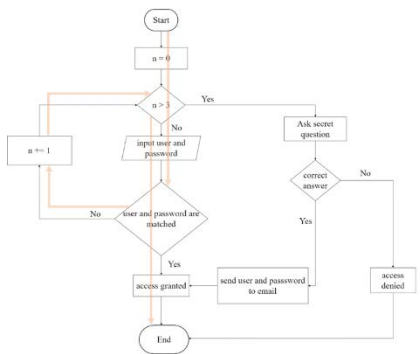
access\_denied()

**ENDIF**

**END**

- Test case

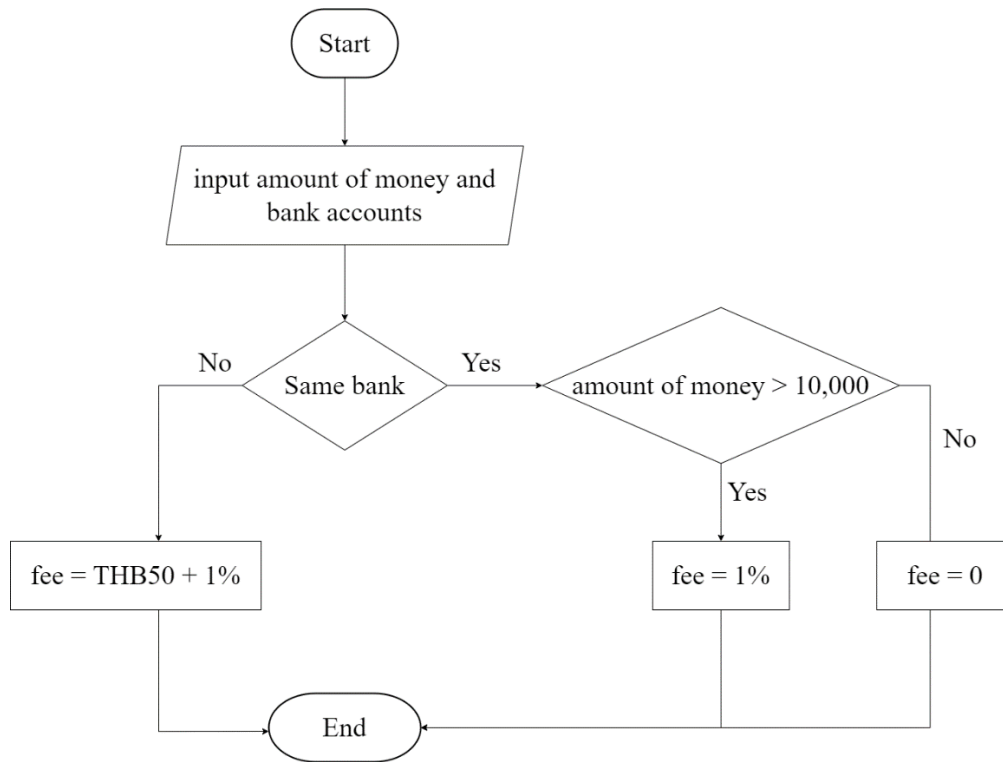
Let User = Hello, Password = 12345 and secret answer = smile

Test case	Inputs	Expected result	Coverage
User and password are matched within 3 times	<ul style="list-style-type: none"><li>▪ User = Hello</li><li>▪ Password = 12345</li></ul>	Access granted	Start → in loop $n > 3 \rightarrow$ access complete → end (Orange) 
User and password are not matched 3 times, but the answer of secret question is correct	<ul style="list-style-type: none"><li>▪ User = Hello</li><li>▪ Password = 00123</li><li>▪ Secret answer = smile</li></ul>	Access granted and user's info will be shown and sent to user's email	Start → in loop for 3 times → ask secret question → access complete → receive user and password → end (Orange → Green)

			<pre> graph TD     Start([Start]) --&gt; n0[n = 0]     n0 --&gt; ngt3{n &gt; 3}     ngt3 -- Yes --&gt; AskSecret[Ask secret question]     ngt3 -- No --&gt; Input[Input user and password]     Input --&gt; Match{user and password are matched}     Match -- No --&gt; ninc1[n += 1]     ninc1 --&gt; ngt3     Match -- Yes --&gt; AccessGranted[access granted]     AccessGranted --&gt; End([End])     AskSecret --&gt; Correct{correct answer}     Correct -- Yes --&gt; SendEmail[send user and password to email]     SendEmail --&gt; End     Correct -- No --&gt; AccessDenied[access denied]     AccessDenied --&gt; End </pre>
<p>User and password are not matched 3 times, but the answer of secret question is incorrect</p>	<ul style="list-style-type: none"> <li>▪ User = Hellp</li> <li>▪ Password = 12345</li> <li>▪ Secret answer = small</li> </ul>	<p>Access denied</p>	<p>Start → in loop for 3 times → ask secret question → access denied → end (Orange → Green)</p> <pre> graph TD     Start([Start]) --&gt; n0[n = 0]     n0 --&gt; ngt3{n &gt; 3}     ngt3 -- Yes --&gt; AskSecret[Ask secret question]     ngt3 -- No --&gt; Input[Input user and password]     Input --&gt; Match{user and password are matched}     Match -- No --&gt; ninc1[n += 1]     ninc1 --&gt; ngt3     Match -- Yes --&gt; AccessGranted[access granted]     AccessGranted --&gt; End([End])     AskSecret --&gt; Correct{correct answer}     Correct -- Yes --&gt; SendEmail[send user and password to email]     SendEmail --&gt; End     Correct -- No --&gt; AccessDenied[access denied]     AccessDenied --&gt; End </pre>

## Scenarios 2

- Flowchart



- Pseudocode

**INPUT:** user enter amount of money (MONEY), user bank account (BANK\_ACCOUNT\_1) and transferred bank account (BANK\_ACCOUNT\_2)

**IF** user bank account (BANK\_ACCOUNT\_1) == transferred bank account (BANK\_ACCOUNT\_2)

**IF** amount of money (MONEY) more than 10,000

        FEE = 1% amount of money (MONEY)

**ELSE**

        FEE = 0

**ENDIF**

**ELSE**

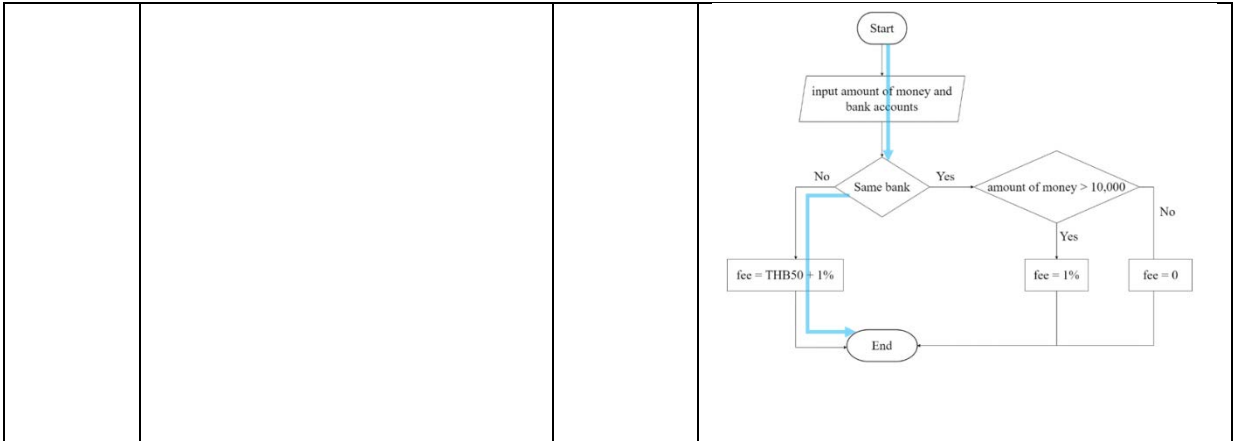
    FEE = THB50 + 1% amount of money (MONEY)

**ENDIF**

**END**

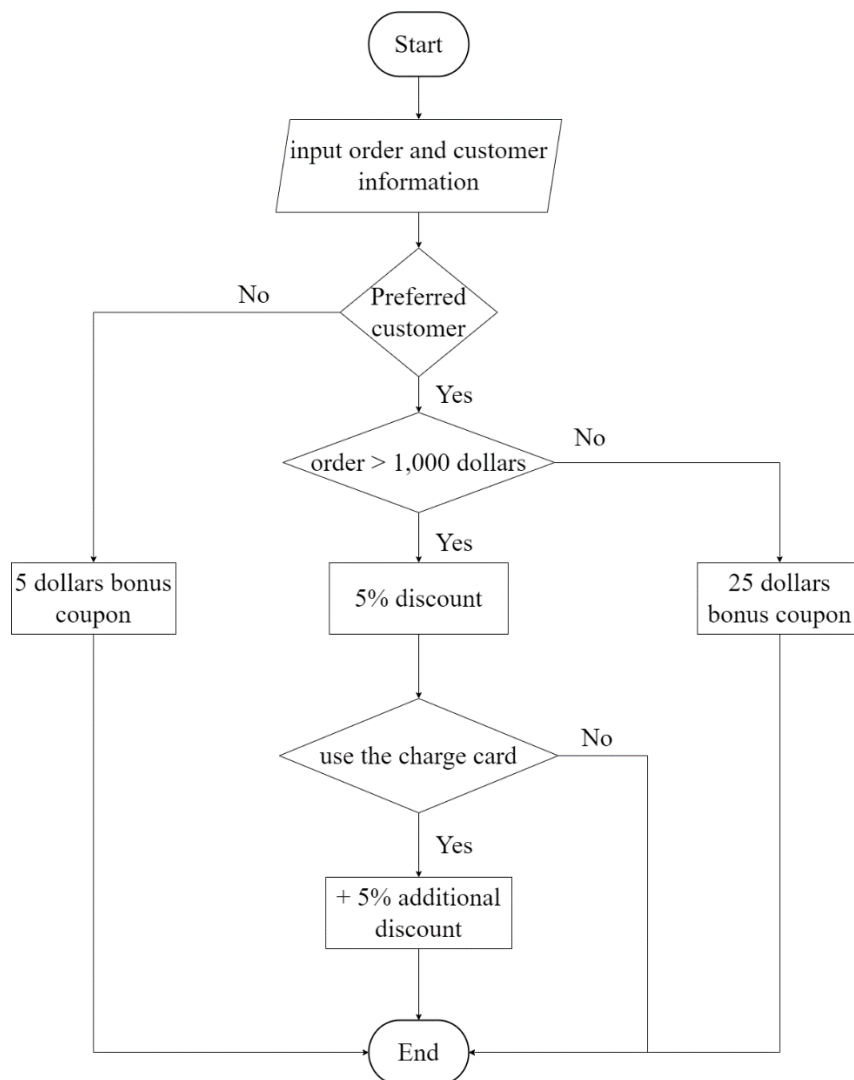
- Test case

Test case	Inputs	Expected results	Coverage
Same bank, but amount of money $\leq$ 10,000	<ul style="list-style-type: none"> <li>▪ User bank account (BANK_ACCOUNT_1) = BANK A</li> <li>▪ Transferred bank account (BANK_ACCOUNT_2) = BANK A</li> <li>▪ Amount of money (MONEY) = 50,000</li> </ul>	fee = 0	<p>Start → receive inputs → check same bank → check amount of money → fee = 0 → end</p> <pre> graph TD     Start([Start]) --&gt; Input[/input amount of money and bank accounts/]     Input --&gt; SameBank{Same bank}     SameBank -- No --&gt; Fee1[fee = THB50 + 1%]     SameBank -- Yes --&gt; AmountMoney{amount of money &gt; 10,000}     AmountMoney -- Yes --&gt; Fee2[fee = 1%]     AmountMoney -- No --&gt; Fee3[fee = 0]     Fee1 --&gt; End([End])     Fee2 --&gt; End     Fee3 --&gt; End   </pre>
Same bank, and amount of money > 10,000	<ul style="list-style-type: none"> <li>▪ User bank account (BANK_ACCOUNT_1) = BANK A</li> <li>▪ Transferred bank account (BANK_ACCOUNT_2) = BANK A</li> <li>▪ Amount of money (MONEY) = 8,000</li> </ul>	fee = 1% amount of money	<p>Start → receive inputs → check same bank → check amount of money → fee = 1% → end</p> <pre> graph TD     Start([Start]) --&gt; Input[/input amount of money and bank accounts/]     Input --&gt; SameBank{Same bank}     SameBank -- No --&gt; Fee1[fee = THB50 + 1%]     SameBank -- Yes --&gt; AmountMoney{amount of money &gt; 10,000}     AmountMoney -- Yes --&gt; Fee2[fee = 1%]     AmountMoney -- No --&gt; Fee3[fee = 0]     Fee1 --&gt; End([End])     Fee2 --&gt; End     Fee3 --&gt; End   </pre>
Different bank	<ul style="list-style-type: none"> <li>▪ User bank account (BANK_ACCOUNT_1) = BANK A</li> <li>▪ Transferred bank account (BANK_ACCOUNT_2) = BANK B</li> <li>▪ Amount of money (MONEY) = 12,000</li> </ul>	fee = THB50 + 1% amount of money	<p>Start → receive inputs → check same bank → fee = THB50 + 1% → end</p> <pre> graph TD     Start([Start]) --&gt; Input[/input amount of money and bank accounts/]     Input --&gt; SameBank{Same bank}     SameBank -- No --&gt; Fee1[fee = THB50 + 1%]     SameBank -- Yes --&gt; AmountMoney{amount of money &gt; 10,000}     AmountMoney -- Yes --&gt; Fee2[fee = 1%]     AmountMoney -- No --&gt; Fee3[fee = 0]     Fee1 --&gt; End([End])     Fee2 --&gt; End     Fee3 --&gt; End   </pre>



### Scenarios 3

- Flowchart



- Pseudocode

**INPUTS:** user enter order (ORDER), customer information (CUTOMER\_INFO), charge card (CHARGE\_CARD)

**IF** customer information (CUSTOMER\_INFO) == preferred customer

**IF** order (ORDER) more than 1,000

        DISCOUNT = 5% of order (ORDER)

**IF** CHARGE\_CARD == use // the customer uses charge card

        DISCOUNT += 5% of order (ORDER)

**ENDIF**

**ELSE**

    25\_coupon() //receive 25 dollars coupon

**ENDIF**

**ELSE**

    5\_coupon() //receive 5 dollars coupon

**ENDIF**

**END**

- Test case

Test case	Inputs	Expected results	Coverage
Preferred customer, order > 1,000 dollars, and customer use charge card	<ul style="list-style-type: none"> <li>▪ user enter order (ORDER) = 1,500</li> <li>▪ customer information (CUTOMER_INFO) = preferred customer</li> <li>▪ charge card (CHARGE_CARD) = use</li> </ul>	5% discount + 5% additional discount = 10% discount	Start → receive inputs → Check preferred customer → Check order amount → receive 5% discount → Check charge card → +5% additional discount → end

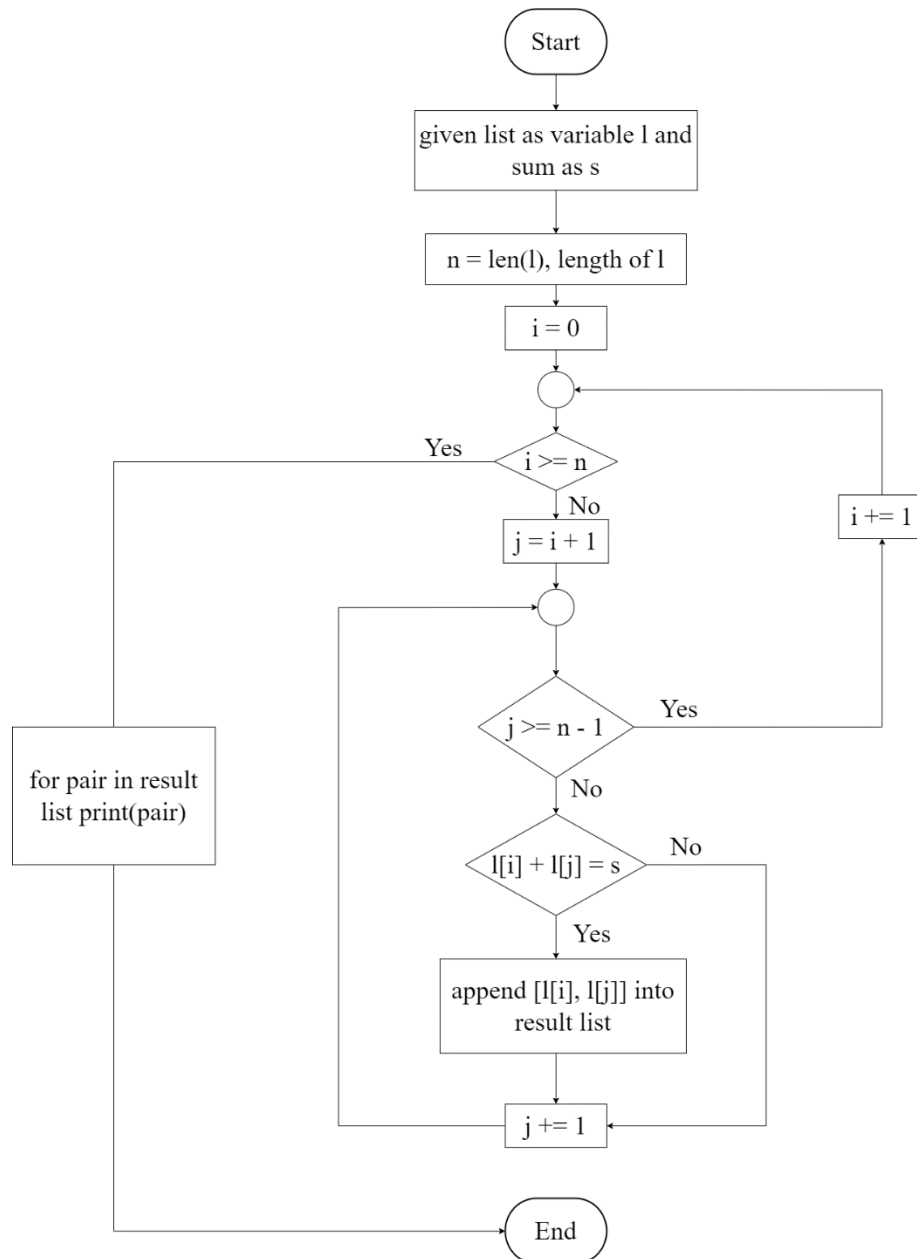
			<pre> graph TD     Start([Start]) --&gt; Input[/input order and customer information/]     Input --&gt; Pref{Preferred customer}     Pref -- No --&gt; B5[5 dollars bonus coupon]     Pref -- Yes --&gt; Order{order &gt; 1,000 dollars}     Order -- No --&gt; B25[25 dollars bonus coupon]     Order -- Yes --&gt; D5[5% discount]     D5 --&gt; Charge{use the charge card}     Charge -- No --&gt; End([End])     Charge -- Yes --&gt; D5p5[+ 5% additional discount]     D5p5 --&gt; End   </pre>
Preferred customer, order > 1,000 dollars, but customer not use charge card	<ul style="list-style-type: none"> <li>▪ user enter order (ORDER) = 1,250</li> <li>▪ customer information (CUTOMER_INF O) = preferred customer</li> <li>▪ charge card (CHARGE_CARD ) = not use</li> </ul>	5% discount	<p>Start → receive inputs → Check preferred customer → Check order amount → receive 5% discount → Check charge card → end</p> <pre> graph TD     Start([Start]) --&gt; Input[/input order and customer information/]     Input --&gt; Pref{Preferred customer}     Pref -- No --&gt; B5[5 dollars bonus coupon]     Pref -- Yes --&gt; Order{order &gt; 1,000 dollars}     Order -- No --&gt; B25[25 dollars bonus coupon]     Order -- Yes --&gt; D5[5% discount]     D5 --&gt; Charge{use the charge card}     Charge -- No --&gt; End([End])     Charge -- Yes --&gt; D5p5[+ 5% additional discount]     D5p5 --&gt; End   </pre>
Preferred customer, but order ≤ 1,000 dollars	<ul style="list-style-type: none"> <li>▪ customer information (CUTOMER_INF O) = preferred customer</li> <li>▪ user enter order (ORDER) = 950</li> </ul>	Receive 25 dollars bonus coupon	<p>Start → receive inputs → Check preferred customer → receive 25 dollars bonus coupon → end</p> <pre> graph TD     Start([Start]) --&gt; Input[/input order and customer information/]     Input --&gt; Pref{Preferred customer}     Pref -- No --&gt; B5[5 dollars bonus coupon]     Pref -- Yes --&gt; Order{order &gt; 1,000 dollars}     Order -- No --&gt; B25[25 dollars bonus coupon]     Order -- Yes --&gt; D5[5% discount]     D5 --&gt; Charge{use the charge card}     Charge -- No --&gt; End([End])     Charge -- Yes --&gt; D5p5[+ 5% additional discount]     D5p5 --&gt; End   </pre>



			<pre> graph TD     Start([Start]) --&gt; Input[/input order and customer information/]     Input --&gt; Pref{Preferred customer}     Pref -- No --&gt; B5[5 dollars bonus coupon]     Pref -- Yes --&gt; Order{order &gt; 1,000 dollars}     Order -- No --&gt; B25[25 dollars bonus coupon]     Order -- Yes --&gt; D5[5% discount]     D5 --&gt; Charge{use the charge card}     Charge -- No --&gt; End([End])     Charge -- Yes --&gt; D5p[+ 5% additional discount]     D5p --&gt; End     B5 --&gt; End     B25 --&gt; End   </pre>
Other customers	<ul style="list-style-type: none"> <li>customer information (CUTOMER_INFO) = not preferred customer</li> </ul>	Receive 5 dollars bonus coupon	<p>Start → receive inputs → Check preferred customer → receive 5 dollars bonus coupon → end</p> <pre> graph TD     Start([Start]) --&gt; Input[/input order and customer information/]     Input --&gt; Pref{Preferred customer}     Pref -- No --&gt; B5[5 dollars bonus coupon]     Pref -- Yes --&gt; Order{order &gt; 1,000 dollars}     Order -- No --&gt; B25[25 dollars bonus coupon]     Order -- Yes --&gt; D5[5% discount]     D5 --&gt; Charge{use the charge card}     Charge -- No --&gt; End([End])     Charge -- Yes --&gt; D5p[+ 5% additional discount]     D5p --&gt; End     B5 --&gt; End     B25 --&gt; End   </pre>

#### Scenarios 4

- Flowchart



- Pseudocode

**SET** given list as  $l$ , given sum as  $s$ , counting position of element in list as  $i$

**SET**  $n$  = length of list

**SET**  $i = 0$

**FOR**  $i < n$

```

j = i + 1
FOR j < n - 1
    IF l[i] + l[j] == s
        Append [l[i], l[j]] into result list
    ENDIF
    Increase value of j by 1
ENDFOR
    Increase value of i by 1
ENDFOR
PRINT pair in result list
END

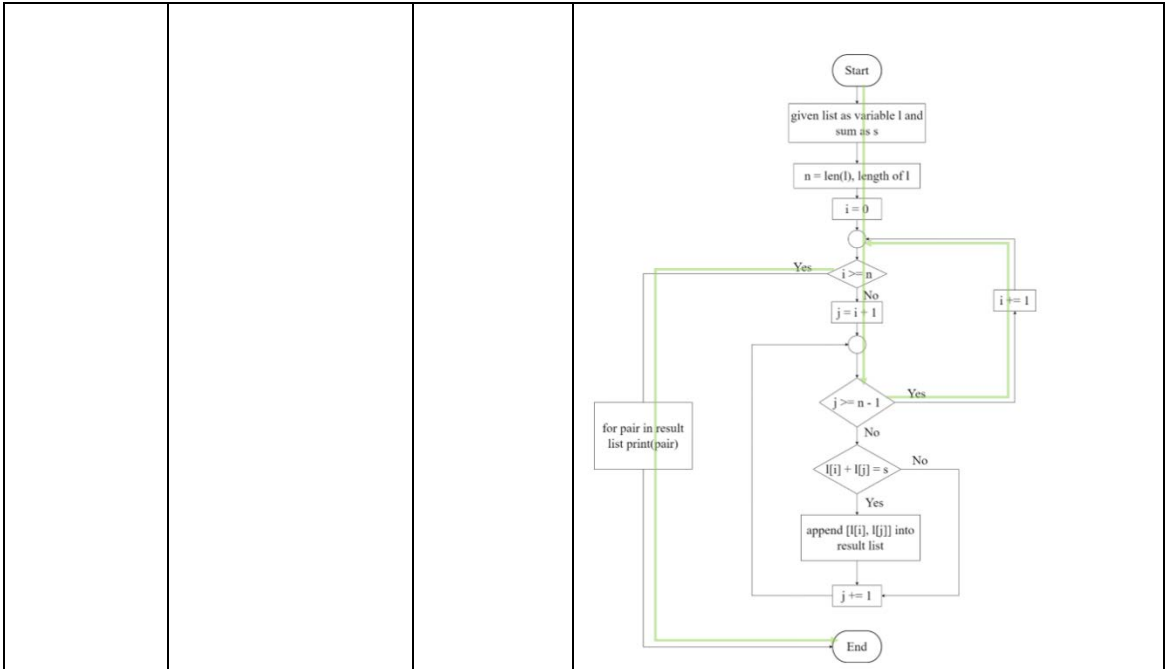
```

- Test case

Test case	Inputs	Expected results	Coverage
Numbers in a given list and the sum are even numbers	<ul style="list-style-type: none"> <li>▪ Given list = [4,6,8,10]</li> <li>▪ Sum = 10</li> </ul>	[4,6]	Start → set variable l,s → set variable n → set variable i → Enter for loop (i < n) → set variable j → Enter another for loop (j < n - 1) → Find the sum of numbers in the given list → Append list of pair that total equal to sum → Reach the last number in list → print result → end

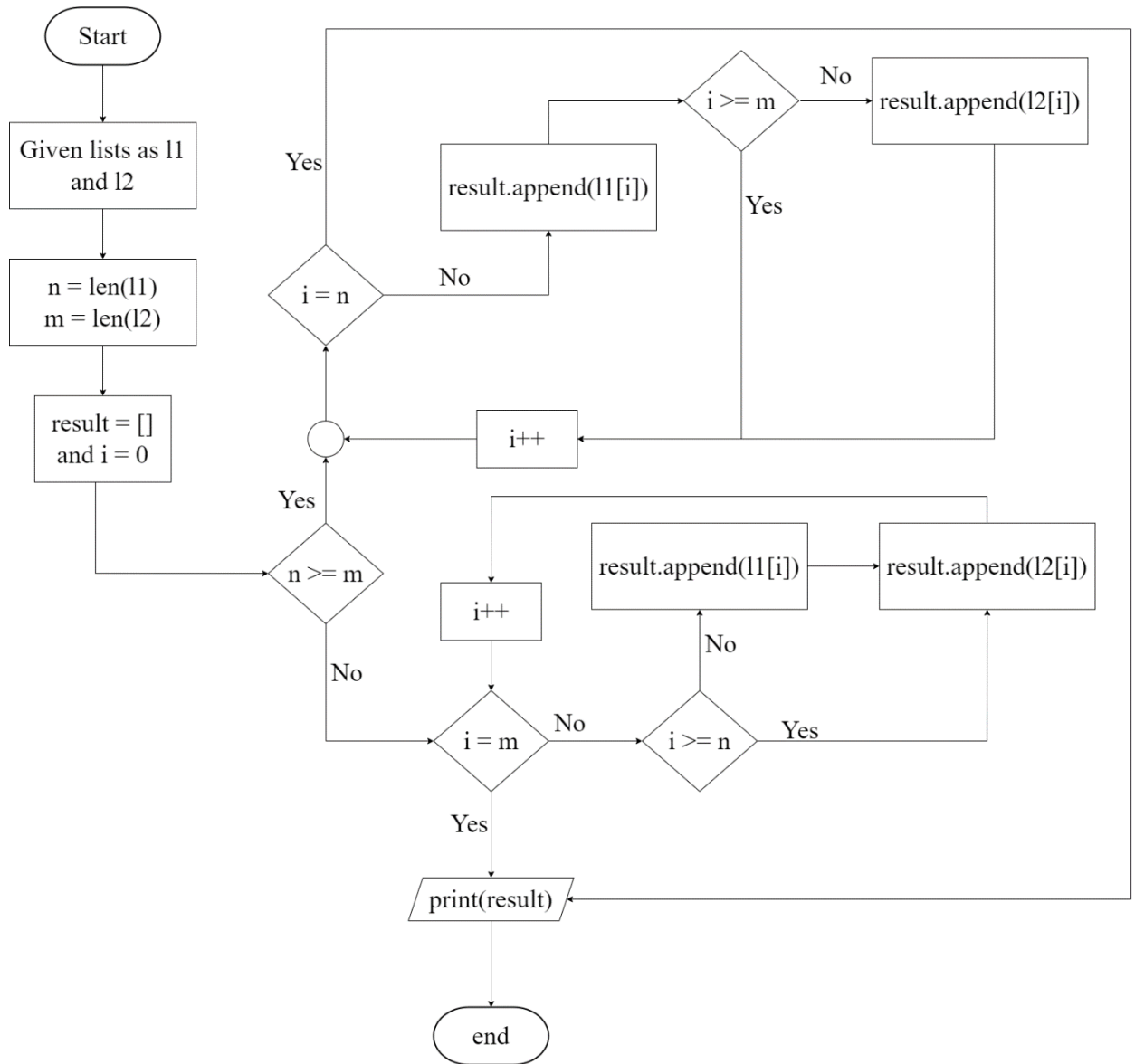
			<pre> graph TD     Start([Start]) --&gt; Init[given list as variable l and sum as s]     Init --&gt; Len[n = len(l), length of l]     Len --&gt; I0[i = 0]     I0 --&gt; Loop1(( ))     Loop1 --&gt; Cond1{i &gt;= n}     Cond1 -- No --&gt; J1[j = i + 1]     J1 --&gt; Loop2(( ))     Loop2 --&gt; Cond2{j &gt;= n - 1}     Cond2 -- No --&gt; Cond3{l[i] + l[j] = s}     Cond3 -- Yes --&gt; Append[append [l[i], l[j]] into result list]     Append --&gt; J2[j += 1]     Cond3 -- No --&gt; J2     Cond2 -- Yes --&gt; I1[i += 1]     J2 --&gt; Loop2     I1 --&gt; Loop1     Cond1 -- Yes --&gt; Print[for pair in result list print(pair)]     Print --&gt; End([End]) </pre>
<p>No pair in a given list that total equal to sum</p>	<ul style="list-style-type: none"> <li>Given list = [1,3,5,7]</li> <li>Sum = 9</li> </ul>	[ ]	<p>Start → set variable l,s → set variable n → set variable i → Enter for loop (i &lt; n) → set variable j → Enter another for loop (j &lt; n – 1) → Find the sum of numbers in the given list → Reach the last number in list → print result → end</p> <pre> graph TD     Start([Start]) --&gt; Init[given list as variable l and sum as s]     Init --&gt; Len[n = len(l), length of l]     Len --&gt; I0[i = 0]     I0 --&gt; Loop1(( ))     Loop1 --&gt; Cond1{i &gt;= n}     Cond1 -- No --&gt; J1[j = i + 1]     J1 --&gt; Loop2(( ))     Loop2 --&gt; Cond2{j &gt;= n - 1}     Cond2 -- No --&gt; Cond3{l[i] + l[j] = s}     Cond3 -- Yes --&gt; Append[append [l[i], l[j]] into result list]     Append --&gt; J2[j += 1]     Cond3 -- No --&gt; J2     Cond2 -- Yes --&gt; I1[i += 1]     J2 --&gt; Loop2     I1 --&gt; Loop1     Cond1 -- Yes --&gt; Print[for pair in result list print(pair)]     Print --&gt; End([End]) </pre>

<p>Repetitive numbers in a given list</p>	<ul style="list-style-type: none"> <li>Given list = [1,3,3,2]</li> <li>Sum = 4</li> </ul>	<p>[1,3], [1,3]</p>	<p>Start → set variable l,s → set variable n → set variable i → Enter for loop (i &lt; n) → set variable j → Enter another for loop (j &lt; n – 1) → Find the sum of numbers in the given list → Append list of pair that total equal to sum → Reach the last number in list → print result → end</p> <pre> graph TD     Start([Start]) --&gt; AssignList[given list as variable l and sum as s]     AssignList --&gt; CalcN[n = len(l), length of l]     CalcN --&gt; InitI[i = 0]     InitI --&gt; Cond1{i &gt;= n}     Cond1 -- Yes --&gt; End([End])     Cond1 -- No --&gt; InitJ[j = i + 1]     InitJ --&gt; Cond2{j &gt;= n - 1}     Cond2 -- Yes --&gt; IncI[i += 1]     IncI --&gt; Cond1     Cond2 -- No --&gt; Cond3{l[i] + l[j] = s}     Cond3 -- Yes --&gt; Append[append [l[i], l[j]] into result list]     Append --&gt; IncJ[j += 1]     Cond3 -- No --&gt; IncJ     IncJ --&gt; PrintPair[for pair in result list print(pair)]     PrintPair --&gt; End   </pre>
<p>Only one number in a given list</p>	<ul style="list-style-type: none"> <li>Given list = [1]</li> <li>Sum = 2</li> </ul>	<p>[ ]</p>	<p>Start → set variable l,s → set variable n → set variable i → Enter for loop (i &lt; n) → set variable j → Enter another for loop (j &lt; n – 1) → Reach the last number in list → print result → end</p>



## Scenarios 5

- Flowchart



- Pseudocode

**SET** list\_1 = l1, list\_2 = l2

**SET** n = len(l1) //length of list\_1

**SET** m = len(l2) //length of list\_2

**SET** result = []

```

SET i = 0 //for position in list
IF length of list_1(n) >= length of list_2(m)
    FOR i < length of list_1(n)
        Append element in list_1(l1) at position i into the result
        IF i >= length of list_2(m)
            pass
        ELSE
            Append element in list_2(l2) at position i into the result
        ENDIF
        Increase value of i by 1
    ENDFOR
ELSE
    FOR i < length of list_2(m)
        IF i >= length of list_1(n)
            Append element in list_2(l2) at position i into the result
        ELSE
            Append element in list_1(l1) at position i into the result
            Append element in list_2(l2) at position i into the result
        ENDIF
        Increase value of i by 1
    ENDFOR
ENDIF
PRINT(result)
END

```



- Test case

Test case	Inputs	Expected Results	Coverage
The lengths of both given lists are equal	<ul style="list-style-type: none"> <li>▪ Given list 1 = [1, 2, 3]</li> <li>▪ Given list 2 = [b, d, e]</li> </ul>	[1, b, 2, d, 3, e]	<p>Start → Set lists as l1 and l2 → set n = length of list 1 and m = length of list 2 → set empty result list and i = 0 → Check n &gt;= m → Enter for loop (i &lt; n) → Append element from list 1 → Check i &gt;= m → Append element from list 2 → increase i by 1 → Append element until reach the last element → print result list → end</p> <pre> graph TD     Start([Start]) --&gt; Init[Given lists as l1 and l2 n = len(l1) m = len(l2)]     Init --&gt; Init2[result = [] i = 0]     Init2 --&gt; Cond1{n &gt;= m}     Cond1 -- Yes --&gt; LoopStart(( ))     LoopStart --&gt; Append1[result.append(l1[i])]     Append1 --&gt; Append2[result.append(l2[i])]     Append2 --&gt; IncI[i++]     IncI --&gt; Cond2{i == n}     Cond2 -- No --&gt; LoopStart     Cond2 -- Yes --&gt; Print[print(result)]     Print --&gt; End([end])     Cond1 -- No --&gt; Print   </pre>
The length of list 1 > the length of list 2	<ul style="list-style-type: none"> <li>▪ Given list 1 = [1, 2, 3, 4]</li> <li>▪ Given list 2 = [b, d]</li> </ul>	[1, b, 2, d, 3, 4]	<p>Start → Set lists as l1 and l2 → set n = length of list 1 and m = length of list 2 → set empty result list and i = 0 → Check n &gt;= m → Enter for loop (i &lt; n) → Append element from list 1 → Check i &gt;= m → Append element from list 2 → increase i by 1 → Append element until reach the last element of list 2 → Append element from list 1 only → reach last element of list 1 → print result list → end</p>

			<pre> graph TD     Start([Start]) --&gt; Init[Given lists as l1 and l2 n = len(l1) m = len(l2) result = [] i = 0]     Init --&gt; D1{n &gt;= m}     D1 -- Yes --&gt; D2{i &gt;= n}     D2 -- Yes --&gt; A1[result.append(l2[i])]     A1 --&gt; I1[i++]     I1 --&gt; D2     D2 -- No --&gt; A2[result.append(l1[i])]     A2 --&gt; I2[i++]     I2 --&gt; D2     D1 -- No --&gt; D3{i &gt;= m}     D3 -- Yes --&gt; A3[result.append(l1[i])]     A3 --&gt; I3[i++]     I3 --&gt; D3     D3 -- No --&gt; A4[result.append(l2[i])]     A4 --&gt; I4[i++]     I4 --&gt; D3     D2 --&gt; Print[/print(result)/]     D3 --&gt; Print     Print --&gt; End([end]) </pre>
<p>The length of list 2 &gt; the length of list 1</p>	<ul style="list-style-type: none"> <li>Given list 1 = [1, 2]</li> <li>Given list 2 = [a, b, c, d]</li> </ul>	<p>[1, a, 2, b, c, d]</p>	<p>Start → Set lists as l1 and l2 → set n = length of list 1 and m = length of list 2 → set empty result list and i = 0 → Check n &gt;= m → Enter for loop (i &lt; m) → Check i &gt;= n → Append element from list 1 then list 2 → increase i by 1 → Append element until reach the last element of list 1 → Append element from list 2 only → reach last element of list 2 → print result list → end</p> <pre> graph TD     Start([Start]) --&gt; Init[Given lists as l1 and l2 n = len(l1) m = len(l2) result = [] i = 0]     Init --&gt; D1{n &gt;= m}     D1 -- Yes --&gt; D2{i &gt;= n}     D2 -- Yes --&gt; A1[result.append(l2[i])]     A1 --&gt; I1[i++]     I1 --&gt; D2     D2 -- No --&gt; A2[result.append(l1[i])]     A2 --&gt; I2[i++]     I2 --&gt; D2     D1 -- No --&gt; D3{i &gt;= m}     D3 -- Yes --&gt; A3[result.append(l1[i])]     A3 --&gt; I3[i++]     I3 --&gt; D3     D3 -- No --&gt; A4[result.append(l2[i])]     A4 --&gt; I4[i++]     I4 --&gt; D3     D2 --&gt; Print[/print(result)/]     D3 --&gt; Print     Print --&gt; End([end]) </pre>