Quine McCluskey Method

총 다섯가지 단계로 구분하였습니다.

첫번째는 minterm input을 받는 단계입니다. 콤마로 구분하여 variable 개수와 상관없이 인풋을 받았습니다.

```
'''Quine McCluskey Method used
1. recieve input of minterm expression
2. convert inputs into binary number
3. determine the prime implicants
4. determine prime implicant chart
5. determine minimum solution using Petrick's method'''

'''step 1. take in input'''
#recieve input of minterm expression devided with commas
minterm_expression = map(int,input("insert your minterm expression (divided with commas):").split(','))
minterms = list(minterm_expression)
```

두번째는 입력된 minterms 들을 이진법 숫자로 변환하였습니다.

Binary notation을 algebraic notation으로 변환하는 convert2letter함수와 각 binary minterm들을 1의 개수에 따라 그룹으로 구분하는 group 함수를 만들었습니다.

```
'''step2. convert inputs into binary number'''
minterm_bin = []

#converting minterm into binary number
for minterm in minterms:
    if max(minterms)<8:
        binary = format(minterm, "03b")
        minterm_bin.append(str(binary))
        var_num = 3
    else:
        binary = format(minterm, "04b")
        minterm_bin.append(str(binary))
        var_num = 4

#function to convert binary number into letter
def convert2letter(num):
    var0 = ['a','b','c','d']
    var1 = ['A','B','C','D']
    letter = ''
    for i in range(len(num)):
        if num[i]=='0':
            letter += var0[i]
        if num[i]=='1':
            letter += var1[i]
    return letter

#dividing minterms in groups
def group(num):
    count = 0
    for i in range(len(num)):
        if num[i]=='1':
            count +=1
    return count
```

세번째 단계는 prime implicants를 결정하는 단계입니다.

우선 각 column을 구하였습니다. Colum1 은 col1 변수에 {그룹:해당 minterm}의 딕셔너리로 저장되었습니다.

```
'''step 3. determine prime implicants'''
#get each column for quine McCluskey method to determine prime implicants
#Column 1
col1 = {0:[], 1:[], 2:[], 3:[], 4:[]}
for i in minterm_bin:
    for n in range(var_num+1):
        if group(i) == n:
            col1[n].append(i)

print(col1,'\n')

col1_val = list(col1.values())
col1_left = []
for x in col1_val:
    for y in x:
        col1_left.append(y)
#terms that are checked
col1_checked = []

```

Colum2는 column1에서 그룹1과 그룹2, 그룹2와 그룹3, 그룹3과 그룹4를 비교하여 묶을 수 있는 항들을 묶었습니다. {묶은 그룹 중 앞의 그룹: 해당 항}으로 저장하였습니다.

그 과정에서 colum1에서 check 되지 않은 항들은 col1_left 리스트에 저장되었습니다.

```
#Column 2
col2 = {0:[], 1:[], 2:[], 3:[]}
for i in range(len(col1)):
    for n in range(len(col1[i])):
        count_left = 0
        for k in range(len(col1[i+1])):
            one_intersect = ''
            count = 0
            for l in range(var_num):
                if col1[i][n][l] == col1[i+1][k][l]:
                    one_intersect += col1[i][n][l]
                else:
                    one_intersect += '-'
                    count +=1
            if count<2:
                col2[i].append(one_intersect)
                col1_checked.append(col1[i][n])
                col1_checked.append(col1[i+1][k])
            list(set(col2[i]))
col1_left = list(set(col1_left)-set(col1_checked))
print('column 1 leftover:', col1_left)
print('column2: ', col2, '\n')


col2_val = list(col2.values())
col2_left = []
for x in col2_val:
    for y in x:
        col2_left.append(y)
col2_checked = []
```

Colum3는 column2에서 그룹1과 그룹2, 그룹2와 그룹3을 비교하여 묶을 수 있는 항들을 묶었습니다. {묶은 그룹 중 앞의 그룹: 해당 항}으로 저장하였습니다.

그 과정에서 colu21에서 check 되지 않은 항들은 col2_left 리스트에 저장되었습니다.

```
#Column 3
col3 = {0:[], 1:[], 2:[]}
for i in range(len(col2)):
    for n in range(len(col2[i])):
        for k in range(len(col2[i+1])):
            two_intersect = ''
            count = 0
            for l in range(var_num):
                if col2[i][n][l] == col2[i+1][k][l]:
                    two_intersect += col2[i][n][l]
                else:
                    two_intersect += '-'
                    count +=1
            if count<2:
                col3[i].append(two_intersect)
                col2_checked.append(col2[i][n])
                col2_checked.append(col2[i+1][k])
            list(set(col3[i]))
col2_left = list(set(col2_left) - set(col2_checked))
print('column 2 leftover: ', col2_left, '\n')
print('column3: ', col3, '\n')


for i in range(len(col3)):
    col3[i] = list(set(col3[i]))

col3_left = []
col3_val = col3.values()
for x in col3_val:
    for y in x:
        col3_left.append(y)
print('column 3 left over: ', col3_left, '\n')

```

prime 리스트는 col1_left, col2_left, col3_left 각 리스트의 합이고, 그 prime 리스트의 각 원소를 convert2letter 함수를 사용하여 algebraic form으로 변환하였습니다.

```

#get prime implicants
prime = []
#convert prime implicants into letter
#lowercase = 0, uppercase = 1
for left1 in col1_left:
    prime.append(convert2letter(str(left1)))
for left2 in col2_left:
    prime.append(convert2letter(str(left2)))
for left3 in col3_left:
    prime.append(convert2letter(str(left3)))
print('prime implicants: ',prime)

```

최소항으로 정리하는 네번째와 다섯번째 단계는 하지 못했습니다.