Switching Theory - CS225

Assignment - 2

Name of Student: M Maheeth Reddy
Roll No.: 1801CS31

Program for Question 7

Aim: To convert standard SOP(sum of products) form to standard POS(product of sums) form
Language: **Python 3**
Program:

```
import re

def forMinIndex(term,var,n):
    split = '|'.join(var)
    boolList = re.split(split,term)
    return boolList

def MinIndex(boolList):
    val = 0
    for i in range(1,len(boolList)):
        bit = 1
        if boolList[i] == "'":
            bit = 0
        val += (2**(len(boolList)-i-1))*bit
    return val

def findMaxVals(MinVals,n):
    MaxVals = []
    MinValsAll = list(range(0,2**n))
    for val in MinValsAll:
        if not val in MinVals:
            MaxVals.append(val)
    for i in range(len(MaxVals)):
        MaxVals[i] = bin(MaxVals[i])[2:]
        while len(MaxVals[i]) != n:
            MaxVals[i] = '0' + MaxVals[i]
    return MaxVals

def bool2MaxVal(binary,var):
    if len(var) == 1:
        return var[0]
    MaxTerm = ''
    for i in range(len(var)):
        MaxTerm = MaxTerm + var[i]
```

```python
        if binary[i] == '1':
            MaxTerm = MaxTerm + "'"
        MaxTerm = MaxTerm+'+'
    MaxTerm = MaxTerm[:-1]
    MaxTerm = '(' + MaxTerm + ')'
    return MaxTerm

def main():
    print('Variables in function should be a,b,c,d,... ')
    n = int(input('Enter no. of variables: '))
    sop = input("Expression in standard SOP form:\n")

    terms = sop.split('+')

    alpha = "abcdefghijklmnopqrstuvwxyz"
    var = []
    for i in range(n):
        var.append(alpha[i])

    MinVals = []
    for term in terms:
        boolList = forMinIndex(term,var,n)
        val = MinIndex(boolList)
        MinVals.append(val)

    MaxVals = findMaxVals(MinVals,n)
    MaxTerms = []
    for val in MaxVals:
        MaxTerms.append(bool2MaxVal(val,var))

    pos = ''.join(MaxTerms)
    print("Expression in standard POS form:")
    print(pos)

if __name__=="__main__":
    main()
```

Input:

```
Variables in function should be a,b,c,d,...
Enter no. of variables: 3
Expression in standard SOP form:
a'b'c'+ab'c+ab'c'+abc
```

Output:

```
Expression in standard POS form:
(a+b+c')(a+b'+c)(a+b'+c')(a'+b'+c)
```

Program for Question 8

Aim: To simplify a 3-variable Boolean expression using K-map

Language: **C++**

Program:

```cpp
#include<iostream>
#include<list>
using namespace std;

int mod(int x);
void printMat(int arr[][4], int row, int col);
void printList(list<int> ls);
void simplify(int kMap[][4], int group[][4], list<int> cubes[]);

int index[] = {0,1,3,2,4,5,7,6};

int main() {
    int minterms,a,b,c,kMap[2][4],group[2][4];
    list<int> cubes[8];
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 4; j++) {
            kMap[i][j] = 0;
            group[i][j] = 0;
        }
    }

    cout << "Variables are a,b,c\n";
    cout << "Give no. of cases of a,b,c when expression(F) is TRUE)\n";
    cin >> minterms;
    if(minterms == 0) {
        cout << "Expression is F = 0\n";
        cout << "K-Map is:\n";
        printMat(kMap,2,4);
        return 0;
    }
    cout << "Values of a b c:\n";
    for(int i = 0; i < minterms; i++) {
        cin >> a >> b >> c;
        kMap[a][mod(3*b-c)] = 1;
    }
    cout << "Input K-Map:\n";
    printMat(kMap,2,4);

    simplify(kMap,group,cubes);
    int noGroups = 0;
    for(int i = 0; i < 8 ; i++)
        if(!cubes[i].empty()) noGroups++;
    cout << "The "<< noGroups << " group(s) formed in the K-Map are:\n";
    for(int i = 0; i < 8 ; i++) {
        if(!cubes[i].empty()) {
            cout << index[i] << " " ;
        printList(cubes[i]);
        }
    }
}

int mod(int x) {
    return (x>0)?x:-x;
}

void printMat(int arr[][4], int row, int col) {
    for(int i = 0; i < row; i++) {
        for(int j = 0; j < col; j++) {
            cout << arr[i][j] << " ";
        }
        cout << "\n";
    }
}

void printList(list<int> ls) {
    list<int> :: iterator it, lastElt = --ls.end();
    for(it = ls.begin(); it != ls.end(); ++it){
        cout << index[*it];
        if(it != lastElt) {
            cout << ' ';
        } else {
            cout << '\n';
        }
    }
}

void simplify(int kMap[][4], int group[][4], list<int> cubes[]) {
    int row1all, row2all;
    for(int i = 0; i < 4 ; i++) {
        if(kMap[0][i] == 1) {
            row1all = 1;
            cubes[0].push_back(i);
            continue;
        }
        row1all = 0;
        cubes[0].clear();
        break;
}

    for(int i = 0; i < 4 ; i++) {
        if(kMap[1][i] == 1) {
            row2all = 1;
            cubes[4].push_back(i+4);
            continue;
        }
        row2all = 0;
        cubes[4].clear();
        break;
    }

    for(int i = 0; i < 4 ; i++) {
        group[0][i] = row1all;
        group[1][i] = row2all;
    }

    if(row1all == 1) cubes[0].pop_front();
    if(row2all == 1) cubes[4].pop_front();
    if(row1all == 1 && row2all == 1) return;

    if(kMap[0][0] == 1 && kMap[0][3] == 1 && row1all == 0) {
        cubes[3].push_back(0);
        group[0][3] = 1;
    }

    if(kMap[1][0] == 1 && kMap[1][3] == 1 && row2all == 0) {
        cubes[7].push_back(4);
        group[1][3] = 1;
    }

    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 3; j++) {
            if(kMap[i][j] == 1 && group[i][j] == 0) {
                if(kMap[i][j+1] == 1) {
                    cubes[4*i+j].push_back(4*i+j+1);
                    group[i][j] = 1;
                }
            }
        }
    }

    for(int i = 0; i < 4 ; i++) {
        if(!cubes[i].empty() && !cubes[i+4].empty()) {
            if(cubes[i].front() == cubes[i+4].front()-4) {
                cubes[i].push_back(i+4);
                cubes[i].push_back(cubes[i+4].front());
                group[0][i] = 1;
                group[0][cubes[i].front()] = 1;
                group[1][i] = 1;
                group[1][cubes[i+4].front()-4] = 1;
                if(cubes[i+4].size() != 4) {
                    cubes[i+4].clear();
                }
            }
        }
    }

    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 4; j++) {
            if(kMap[i][j] == 1 && group[i][j] == 0) {
                if(kMap[!i][j] == 1) {
                    cubes[4*i+j].push_back(4*(!i)+j);
                    group[!i][j] = 1;
                } else if(j != 0 || kMap[i][3] != 1) {
                    cubes[4*i+j].push_back(4*i+j);
                }
                group[i][j] = 1;
            }
        }
    }
}
```

Input 1:

```
Variables are a,b,c
Give no. of cases of a,b,c when expression(F) is TRUE)
4
Values of a b c:
0 0 0
0 1 0
0 0 1
1 1 1
```

Output 1:

```
Input K-Map:
1 1 0 1
0 0 1 0
The 4 group(s) formed in the K-Map are:
0 1
1 1
2 0
7 7
```

Input 2:

```
Variables are a,b,c
Give no. of cases of a,b,c when expression(F) is TRUE)
8
Values of a b c:
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
```

Output 2:

```
Input K-Map:
1 1 1 1
1 1 1 1
The 2 group(s) formed in the K-Map are:
0 1 3 2
4 5 7 6
```