**PROJECT REPORT**

**Paper: Programming using C**

**Code: MCA-PG-C104**

**Subject: MCA**

**Semester: I**

**Department of Computer Applications**

****

**Sikkim University**

**[November, 2020 - April, 2021]**

**Name of Student: RAM BABU RAY**

**Roll No: 20MCA013**

**Student’s Signature:**

**Course Faculty: Dr. Swarup Roy**

**Designation: Head of Department of Computer Applications, Sikkim University**

**Project Title:** BOOLEAN EQUATION SIMPLIFIER

#### Language to be used: C

**Brief Description:**

We created a Boolean equation simplifier, which takes a Boolean function as input, and gives minimized function.

**Compatibility:**

* turoC

**Features implemented:**

* a manual is added in the program to instruct, in which format input should be given.
* Program is performing Quine–McCluskey formula on given equation for minimization.
* **INTRODUCTION:**

For minimizing the circuit, we use different minimization techniques.

Minimizing a circuit will integrate the circuit, decreases the required number of gates and also reduces the cost.

In this project, user have to input their Boolean equation in the form 0’s and 1’s for example: for X’YZ+XYZ, user have to write 011+111. program will read this equation in a string. Program will perform Quine–McCluskey formula on it, and it will print minimized result.

**ALGORITHM**

**Case 1:// printing manual**

**Step 1: print** 1.THIS PROGRAM CAN PERFORM ONLY SOP MINIMIZATION:

**Step2: print** 2.ENTER THE EQUATION IN CANONICAL FORM:

**Step3: print** 3.ENTER THE EQUATION IN SUM OF BINARY NUMBER'S FORM

EG: FOR EQUATION XYZ+X'Y'Z'\n\t\tINSERT 111+000

**Step4: print** 4.OUTPUT WILL ALSO DISPLAYED IN SIMILAR FORM:

**Step5: print** EG FOR Y'Z'+X'Z'\n\t PROGRAM WILL PRINT \_00+0\_0

**Step6: print** \*\*\*\*\*\*\*\*\*\*\*\*\*PRESS ENTER TO PROCEED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Case 2:// preforming** **Quine–McCluskey**

**Step 1:** read the function in variable a.

**Step 2:** declare a structure named boo

**Step 2.1:** declare a char variable ab[10]

**Step 2.2:** declare a int variable p

**Step 3:** declare a function **var** to countsize of single element.

**Step 3.1:**  loop from i=0 condition i<l updating i=i+1

{

**Step3.1.1** if(a[i]=='+')

**Step 3.1.1.1** return(i);

}

**Step 3.2** return(0);

**Step 4:** l= length of variable a

**Step 5:** d= **var(a)**

**Step 5.1:** goto step 3 with value a.

**Step 6:** N=l/d

**Step 7:** dynamically declaring variable p of size (d+1).

**Step 8:** p[0]=0

**Step 9:** dynamically declaring a 3D array ‘A’ variable of structure boo type

(d+1) X (d+1) X (n\*2)

**Step 11:** initializing each element A[][][].ab[]=NULL and A[][][].p=-1

**Step 12:** loop from k=0 condition k<l updating k=k+(d+1))

{ **step 12.1** loop from j=0 condition j<d updating i=i+1

{ **step 12.1.1** B.ab[j]=a[j+k];

**step 12.1** .**2** if(a[j+k]=='1')

**step 12.1.2.1** count++

}

**Step 12.2:** L=p[count]

**Step 12.3:** B.ab[j]=NULL;

**Step 12.4:** A[0][count][L].p=0;

**Step 12.5:** copy content of B.ab to A[0][count][L].ab.

**Step 12.6:** p[count]++;

**Step 12.7:** count=0;

}

**Step 13:** Q=0;

**Step 14:** loop from x=0 condition x<d updating x=x+1

{

**Step 14.1** loop from i2=0 condition i2<=d increment i2=i2+1

**Step14.1.1** p[i2]=0;

**Step 14.2** loop from i=0 condition i<d-x increment i=i+1

{

**Step 14.2.1** loop from j=0 condition j<N\*2 updating j=j+1

{

**Step 14.2.1.1** loop from k=0 condition k<N\*2 updating k=k+1

{

**Step 14.2.1.1.1** count=0;

**Step 14.2.1.1.2** loop from i1=0 condition i1<d updating i1=i1+1

{

**Step 14.2.1.1.2.1** if(A[x][i][j].ab[i1]==A[x][i+1][k].ab[i1] && A[x][i][j].ab[i1]!=NULL)

{

**Step 14.2.1.1.2.1.** count++;

}

}

**Step 14.2.1.1.3** if(count==d-1)

{

**Step 14.2.1.1.3.1** L=p[i]; temp=1;

**Step 14.2.1.1.3.2** A[x][i][j].p=1;

**Step 14.2.1.1.3.3** A[x][i+1][k].p=1;

**Step 14.2.1.1.3.4** loop from i1=0 condition i1<d updating i1=i1+1

{

**Step 14.2.1.1.3.4.1** if(A[x][i][j].ab[i1]==A[x][i+1][k].ab[i1])

**Step 14.2.1.1.3.4.1.1** A[x+1][Q][L].ab[i1]=A[x][i][j].ab[i1];

else

**Step 14.2.1.1.3.4.1.2** A[x+1][Q][L].ab[i1]='\_';

}

**Step 14.2.1.1.3.4.2** p[i]++;

**Step 14.2.1.1.3.4.3** A[x+1][Q][L].ab[i1]=NULL;

**Step 14.2.1.1.3.4.4** A[x+1][Q][L].p=0;

}

}

}

**Step 14.1.2** if(temp==1)

**Step 14.1.2.1** Q++;temp=0;

}

**Step 14.2** Q=0;

**Step 14.3** temp=0;

}

**Step 15** declaring a variable char val[100][10], variable type int M=0, variable type int temp=0

**Step 16:** loop from i=0 condition i<=d updating i=i+1)

**Step 16.1:** loop from j=0 condition j<=d-I updating j=j+1

**Step 16.1.1:** loop from k=0 condition k<N\*2 updating k=k+1

{

**Step 16.1.1.1:** if(A[i][j][k].p==0)

{

**Step 16.1.1.1.1:** copy content of A[i][j][k].ab to val[M]

**Step 16.1.1.1.2:**M=M+1;

}

}

**Step 17:** Print “result="

**Step 18:** loop from i=0 condition i<M updating i=i+1

{

**Step 18.1:**  Loop from j=i+1 condition j<M updating j=j+1

**Step 18.1.1:**  if(!strcmp(val[i],val[j]))

**Step 18.1.1.1:**  temp=1;

**Step 18.2:** if(temp==0)

**Step 18.2.2:** print val[i]

**Step 18.3:**  temp=0;

}

* **BACKGROUND WORKING PROCESS**

LET INSERTED EQUATION BE:

0000+0001+0010+1000+1010+1011+1110+1111

**ITERATION 1: A[0]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0000  P=0 | 0001  P=0 | 1010  P=0 | 1011  P=0 | 1111  P=0 |
| P=-1 | 0010  P=0 | P=-1 | 1110  P=0 | 1111  P=0 |
| P=-1 | 1000  P=0 | P=-1 | P=-1 | P=-1 |
| P=-1 | P=-1 | P=-1 | P=-1 | P=-1 |
| P=-1 | P=-1 | P=-1 | P=-1 | P=-1 |

**ITERATION 2: A[1]**

|  |  |  |  |
| --- | --- | --- | --- |
| 000\_  P=0 | \_010  P=0 | 101\_  P=0 | 1\_11  P=0 |
| 00\_0  P=0 | 10\_0  P=0 | 1\_10  P=0 | 111\_  P=0 |
| \_000  P=0 | P=-1 | P=-1 | P=-1 |
| P=-1 | P=-1 | P=-1 | P=-1 |
| P=-1 | P=-1 | P=-1 | P=-1 |

**ITERATION 3: A[2]**

|  |  |  |
| --- | --- | --- |
| \_0\_0  P=0 | 1\_1\_  P=0 | P=-1 |
| \_0\_0  P=0 | 1\_1\_  P=0 | P=-1 |
| P=-1 | P=-1 | P=-1 |
| P=-1 | P=-1 | P=-1 |
| P=-1 | P=-1 | P=-1 |

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<alloc.h>

struct boo

{char ab[10];

int p;

};

void read(char a[])

{ printf("enter your expression in sop:\n");

scanf("%s",a);

}

int var(char a[],int l)

{

for(int i=0;i<l;i++)

{ if(a[i]=='+')

return(i);

}

return(0);

}

void sop(char a[])

{int i,x,i1,i2,j,L,M=0,\*p,l,Q=0,d,N,count=0,k,temp=0;

l=strlen(a);

d=var(a,l);

N=l/d;

p=(int\*)malloc((d+1)\*sizeof(int));

for(i=0;i<=d;i++)

p[i]=0;

//printf("value of d is=%d, l=%d",d,l);

struct boo \*\*\*A,B;

A=(boo\*\*\*)malloc((d+1)\*sizeof(boo));

for(i=0;i<=d;i++)

A[i]=(boo\*\*)malloc((d+1-i)\*sizeof(boo));

for(i=0;i<=d;i++)

for(j=0;j<=d-i;j++)

A[i][j]=(boo\*)malloc(N\*2\*sizeof(boo));

for(i=0;i<=d;i++)

for(j=0;j<=d-i;j++)

for(k=0;k<N\*2;k++)

{ A[i][j][k].p=-1;

for(M=0;M<10;M++)

A[i][j][k].ab[M]=NULL;

}

//part 1

for(k=0;k<l;k+=(d+1))

{ for(j=0;j<d;j++)

{B.ab[j]=a[j+k];

if(a[j+k]=='1')

count++;

}// printf("\ncount=%d\n",count);

L=p[count];

B.ab[j]=NULL;

A[0][count][L].p=0;

strcpy(A[0][count][L].ab,B.ab);

// printf("%s",A[L][count].ab);

p[count]++;

count=0;

}

Q=0;

//part2

for(x=0;x<d;x++)

{

for(i2=0;i2<=d;i2++)

p[i2]=0;

for(i=0;i<d-x;i++)

{ for(j=0;j<N\*2;j++)

{

for(k=0;k<N\*2;k++)

{//compare A[0][i][j]==A[0][i+1][k];

count=0;

for(i1=0;i1<d;i1++)

{ if(A[x][i][j].ab[i1]==A[x][i+1][k].ab[i1] && A[x][i][j].ab[i1]!=NULL)

{ count++;

}

}

if(count==d-1)

{ L=p[i]; temp=1;

A[x][i][j].p=1;

A[x][i+1][k].p=1;

for(i1=0;i1<d;i1++)

{ if(A[x][i][j].ab[i1]==A[x][i+1][k].ab[i1])

{ A[x+1][Q][L].ab[i1]=A[x][i][j].ab[i1];

}

else

{ A[x+1][Q][L].ab[i1]='\_';

}

}

p[i]++;

A[x+1][Q][L].ab[i1]=NULL;

A[x+1][Q][L].p=0;

}

}

}

if(temp==1)

{Q++;temp=0;}

}

Q=0;

temp=0;

}

printf("\n");

char val[100][10];

M=0; temp=0;

for(i=0;i<=d;i++)

for(j=0;j<=d-i;j++)

for(k=0;k<N\*2;k++)

{

if(A[i][j][k].p==0)

{strcpy(val[M],A[i][j][k].ab);

M++;}

}

printf("result=");

for(i=0;i<M;i++)

{ for(j=i+1;j<M;j++)

if(!strcmp(val[i],val[j]))

temp=1;

if(temp==0)

printf("%s, ",val[i]);

temp=0;

}

}

void main()

{ clrscr();

char \*a;

int c,s;

printf("enter 1 to read manual:\nenter 2 to proceed:\n");

scanf("%d",&s);

if(s==2)

{ raavan:

clrscr();

read(a);

sop(a);

getch();

}

else

{clrscr();

printf("1.THIS PROGRAM CAN PERFORM ONLY SOP MINIMIZATION:\n");

printf("2.ENTER THE EQUATION IN CANONICAL FORM:\n");

printf("3.ENTER THE EQUATION IN SUM OF BINARY NUMBER'S FORM\nEG: FOR EQUATION XYZ+X'Y'Z'\n\t\tINSERT 111+000\n");

printf("4.OUTPUT WILL ALSO DISPLAYED IN SIMILAR FORM:\n");

printf("EG FOR Y'Z'+X'Z'\n\t PROGRAM WILL PRINT \_00+0\_0");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*PRESS ENTER TO PROCEED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

getch();

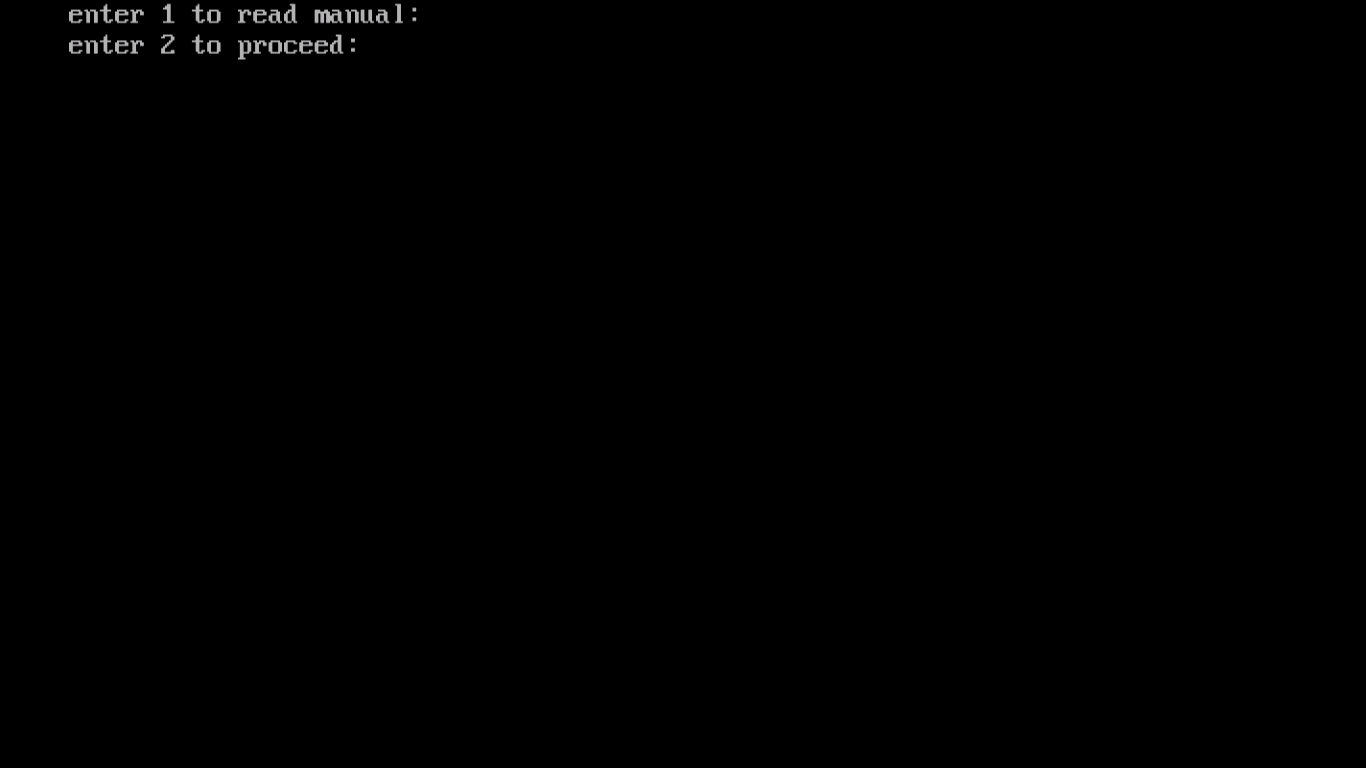
goto raavan;

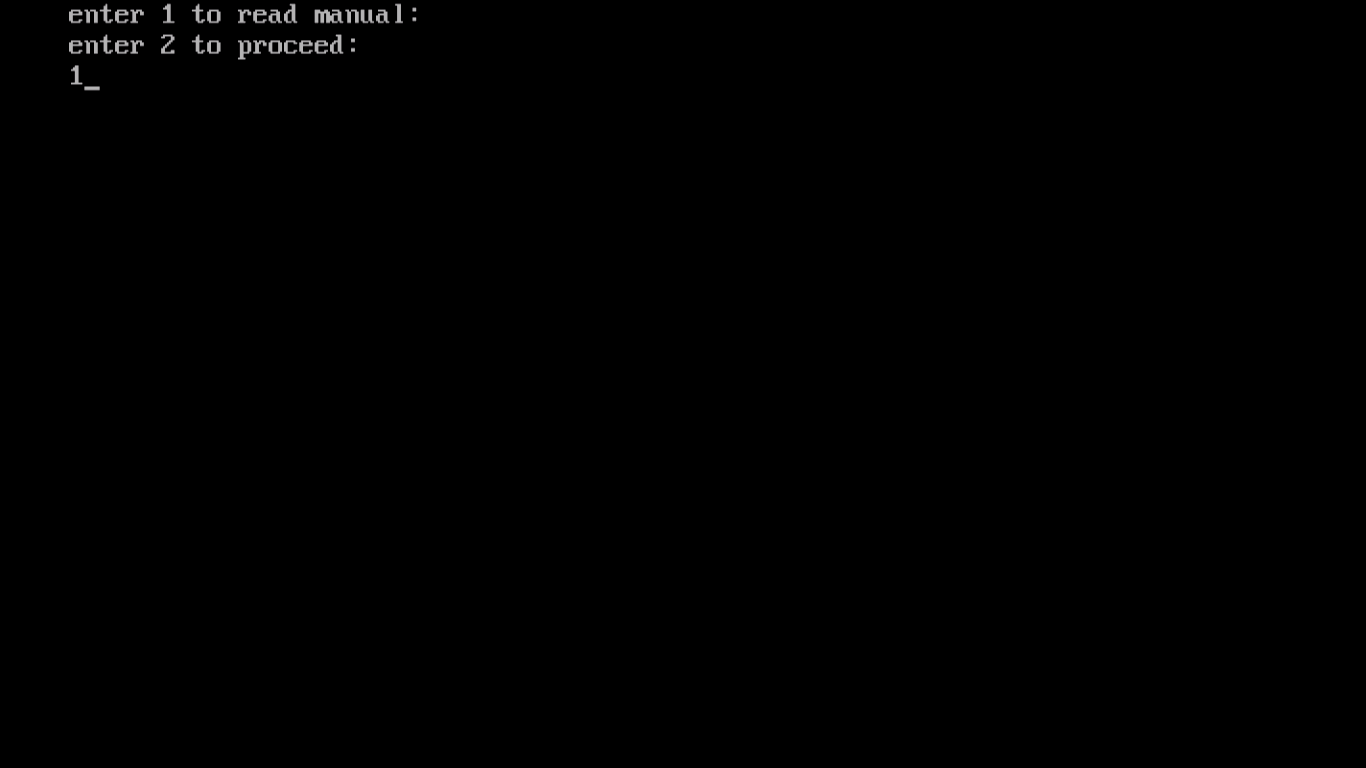
}

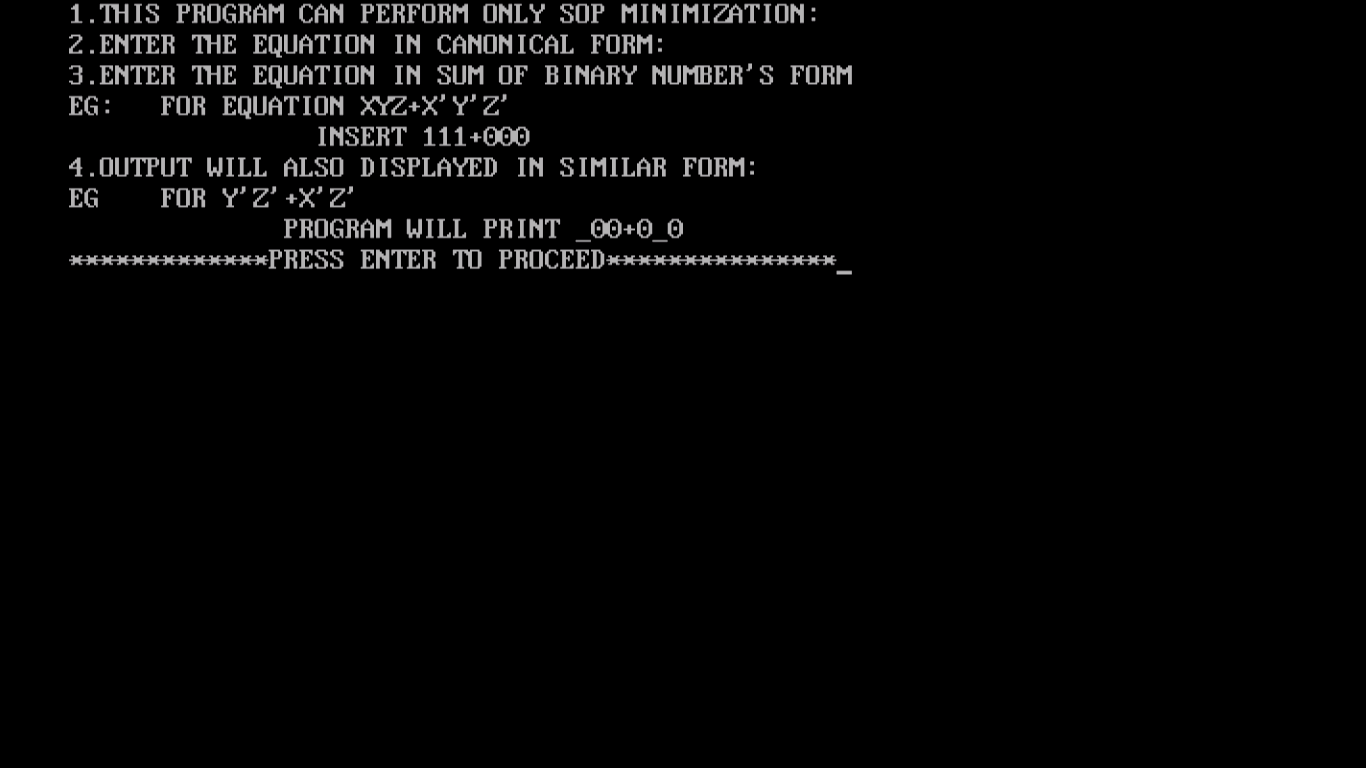
getch();

}

**OUTPUT:**

**STEP 1:**

**STEP 2:**

**STEP 3:**

**STEP 4:**

**STEP 5:**