The C-Code:

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  FILE *fin,*fout;
4  int recStack[200][9],n,funcStack[200];
5  long cnt=0;
6  int adj[9][9];
7  int check[16777216];
8  long long oct[8]={1,8,64,512,4096,32768,262144L,2097152L};
9  long long hash,hash_max=-1;
10 int min=1;
11 int blkComm(int i,int level)
12 {
13         int l=-1,j=level-1;
14         while(j>=1 && (adj[i][funcStack[j]]==0))
15         {
16                 if(l < funcStack[j])l=funcStack[j];
17                 j--;
18         }
19         if(l>i) return 1;
20         else if(j>=2 && (i==funcStack[j-1]) && (i > funcStack[j]) && (adj[i][
               funcStack[j]]==1)) return 1;
21         else return 0;
22 }
23 void DFS(int level)
24 {
25         int flagt=0;
26         for(int z=1;z<=n;z++)
27                 if(recStack[level-1][z]>0) flagt=1;
28         if(flagt==0)
29         {
30                 hash=0;
31                 for(int j=1;j<=8;j++)
32                 {
33                         hash=hash+(oct[j-1]*(recStack[level-1][j]+7));
34                 }
35                 if(check[hash]==0)
36                 {
37                         cnt++;
38                         check[hash]=1;
39                         for(int j=1;j<=8;j++) fprintf(fout,"%d ",recStack[
                               level-1][j]);
40                         fprintf(fout,"\n");
41                         for(int k=1;k<level;k++) fprintf(fout,"s%d ",funcStack
                               [k]);
42                         fprintf(fout,"\n");
43                 }
44                 return;
45         }
46         int flag=0;
```

```
47          for(int i=1;i<=n;i++)
48          {
49                  if(recStack[level-1][i]>0 && (i != funcStack[level-1]) && (
                        blkComm(i,level)==0) && !(level>1 && (i==funcStack[level
                        -2]) && (i>funcStack[level-1])))
50                  {
51                          if((i==5 || i==6 || i==7) && (recStack[level-1][i-1] +
                                recStack[level-1][i+1] < 2*recStack[level-1][i]))
52                          {
53                                  funcStack[level]=i;
54                                  for(int j=1;j<=n;j++)
55                                  {
56                                          if(j!=i) recStack[level][j]=recStack[
                                                level-1][j];
57                                          else recStack[level][j]= recStack[
                                                level-1][j-1] + recStack[level-1][
                                                j+1]- recStack[level-1][j];
58                                  }
59                                  DFS(level+1);
60                                  flag=1;
61                          }
62                          else if(i==1 && (recStack[level-1][3]<2*recStack[level
                                -1][1]))
63                          {
64                                  funcStack[level]=1;
65                                  for(int j=1;j<=n;j++)
66                                  {
67                                          if(j!=1) recStack[level][j]=recStack[
                                                level-1][j];
68                                          else recStack[level][j]= recStack[
                                                level-1][3] - recStack[level-1][j
                                                ];
69                                  }
70                                  DFS(level+1);
71                                  flag=1;
72                          }
73                          else if(i==8 && (recStack[level-1][7]<2*recStack[level
                                -1][8]))
74                          {
75                                  funcStack[level]=8;
76                                  for(int j=1;j<=n;j++)
77                                  {
78                                          if(j!=8) recStack[level][j]=recStack[
                                                level-1][j];
79                                          else recStack[level][j]= recStack[
                                                level-1][7] - recStack[level-1][j
                                                ];
80                                  }
81                                  DFS(level+1);
82                                  flag=1;
83                          }
```

```
84                          else if(i==2 && (recStack[level-1][4]<2*recStack[level
                                -1][2]))
85                          {
86                                  funcStack[level]=2;
87                                  for(int j=1;j<=n;j++)
88                                  {
89                                          if(j!=2) recStack[level][j]=recStack[
                                                level-1][j];
90                                          else recStack[level][j]= recStack[
                                                level-1][4] - recStack[level-1][j
                                                ];
91                                  }
92                                  DFS(level+1);
93                                  flag=1;
94                          }
95                          else if(i==4 && (recStack[level-1][3] + recStack[level
                                -1][5] + recStack[level -1][2]<2*recStack[level
                                -1][4]))
96                          {
97                                  funcStack[level]=4;
98                                  for(int j=1;j<=n;j++)
99                                  {
100                                         if(j!=4) recStack[level][j]=recStack[
                                                level-1][j];
101                                         else recStack[level][j]= recStack[
                                                level-1][3] + recStack[level-1][5]
                                                 + recStack[level -1][2] -
                                                recStack[level-1][j];
102                                 }
103                                 DFS(level+1);
104                                 flag=1;
105                         }
106                         else if(i==3 && (recStack[level-1][1] + recStack[level
                                -1][4]<2*recStack[level-1][3]))
107                         {
108                                 funcStack[level]=3;
109                                 for(int j=1;j<=n;j++)
110                                 {
111                                         if(j!=3) recStack[level][j]=recStack[
                                                level-1][j];
112                                         else recStack[level][j]= recStack[
                                                level-1][1] + recStack[level
                                                -1][4]- recStack[level-1][j];
113                                 }
114                                 DFS(level+1);
115                                 flag=1;
116                         }
117                 }
118         }
119 }
120 int main()
```

```
{
        fin = fopen("data","r");
        fout = fopen("output","w");
        adj[1][3]=adj[3][1]=1;
        adj[2][4]=adj[4][2]=1;
        adj[3][4]=adj[4][3]=1;
        adj[4][5]=adj[5][4]=1;
        adj[6][5]=adj[5][6]=1;
        adj[6][7]=adj[7][6]=1;
        adj[8][7]=adj[7][8]=1;
        while(fscanf(fin,"%d",&n)!=EOF){
                        int i;
                        for(i=1;i<=n;i++) fscanf(fin,"%d",&recStack[0][i]);
                        DFS(1);
                        fprintf(fout,"\n
                            ****************************************\n");
        }
        printf("%d",cnt);
        fclose(fin);
        fclose(fout);
}
```

```
#include<iostream>
#include<string>

using namespace std;

string st1,st2,S[9999],T[9999];
int curr1,curr2,Minimal[9999],n;

bool isSubSequence(string s1, string s2)   //Returns True if s1 is a sub-
    sequence of s2. Returns False otherwise.
{
  curr2=0;
  curr1=0;

  while(1)
  {
    if(curr1 == s1.length())
      return true;

    if(curr2 == s2.length())
      return false;

    if(s2[curr2]!=s1[curr1])
    {
      curr2++;
    }
    else
    {
      curr2++;
      curr1++;
```

```
30        }
31     }
32  }
33
34
35  int main()
36  {
37     /*
38     cin>>n;
39     for(int i=1;i<=n;i++)
40     {
41        getline(cin,S[i]);
42        while(S[i].length()==0)
43           getline(cin,S[i]);
44     }
45     */
46
47     int i=1;
48     while(getline(cin,T[i]))
49     {
50        getline(cin,S[i]);
51  //        while(S[i].length()==0)
52  //           getline(cin,S[i]);
53        i++;
54     }
55
56     n=i-1;
57     /*
58     for(int i=1;i<=n;i++)
59        cout<<S[i]<<"\n\n";
60     */
61
62
63     for(int i=1;i<=n;i++)
64     {
65        Minimal[i]=1;
66        for(int j=1;j<=n;j++)
67        {
68           if(j==i)
69              continue;
70           if(isSubSequence(S[j],S[i]))
71           {
72              Minimal[i]=0;
73              break;
74           }
75        }
76     }
77
78
79     cout<<"The Minimal Words are:\n";
80     for(int i=1;i<=n;i++)
```

```
81      {
82          if(Minimal[i]==1)
83              cout<<T[i]<<"\n"<<S[i]<<"\n";
84      }
85
86
87  }
```