<u>The C-Code:</u>

```
#include<stdio.h>
#include<stdlib.h>
FILE *fin,*fout;
int recStack[200][9],n,funcStack[200];
long cnt=0;
int adj[9][9];
int check[16777216];
long long oct[8]={1,8,64,512,4096,32768,262144L,2097152L};
long long hash,hash_max=-1;
int min=1;
int blkComm(int i,int level)
{
        int l=-1,j=level-1;
        while(j>=1 && (adj[i][funcStack[j]]==0))
        {
                if(l < funcStack[j]) l=funcStack[j];
                j--;
        }
        if(l>i) return 1;
        else if(j>=2 && (i==funcStack[j-1]) && (i > funcStack[j]) && (adj[i][funcStack[j]]==1)) return 1;
        else return 0;
}
void DFS(int level)
{
        int flagt=0;
        for(int z=1;z<=n;z++)
                if(recStack[level-1][z]>0) flagt=1;
        if(flagt==0)
        {
                hash=0;
                for(int j=1;j<=8;j++)
                {
                        hash=hash+(oct[j-1]*(recStack[level-1][j]+7));
                }
                if(check[hash]==0)
                {
                        cnt++;
                        check[hash]=1;
                        for(int j=1;j<=8;j++) fprintf(fout,"%d␣",recStack[level-1][j]);
                        fprintf(fout,"\n");
                        for(int k=1;k<level;k++) fprintf(fout,"s%d␣",funcStack[k]);
                        fprintf(fout,"\n");
                }
                return;
        }
        int flag=0;
        for(int i=1;i<=n;i++)
        {
            if(recStack[level-1][i]>0 && (i != funcStack[level-1]) && (blkComm(i,level)==0) && !(level>1 && (i==funcStack[level-2]) && (i>funcStack[level-1])))
            {
                if((i==5 || i==6 || i==7) && (recStack[level-1][i-1] + recStack[level-1][i+1] < 2*recStack[level-1][i]))
                {
                    funcStack[level]=i;
                    for(int j=1;j<=n;j++)
                    {
                        if(j!=i) recStack[level][j]=recStack[level-1][j];
                        else recStack[level][j]= recStack[level-1][j-1] + recStack[level-1][j+1]- recStack[level-1][j];
                    }
                    DFS(level+1);
                    flag=1;
                }
                else if(i==1 && (recStack[level-1][3]<2*recStack[level-1][1]))
                {
                    funcStack[level]=1;
                    for(int j=1;j<=n;j++)
                    {
                        if(j!=1) recStack[level][j]=recStack[level-1][j];
                        else recStack[level][j]= recStack[level-1][3] - recStack[level-1][j];
                    }
                    DFS(level+1);
                    flag=1;
                }
                else if(i==8 && (recStack[level-1][7]<2*recStack[level-1][8]))
                {
                    funcStack[level]=8;
                    for(int j=1;j<=n;j++)
                    {
                        if(j!=8) recStack[level][j]=recStack[level-1][j];
                        else recStack[level][j]= recStack[level-1][7] - recStack[level-1][j];
                    }
                    DFS(level+1);
                    flag=1;
                }
                else if(i==2 && (recStack[level-1][4]<2*recStack[level-1][2]))
                {
                    funcStack[level]=2;
                    for(int j=1;j<=n;j++)
                    {
                        if(j!=2) recStack[level][j]=recStack[level-1][j];
                        else recStack[level][j]= recStack[level-1][4] - recStack[level-1][j];
                    }
                    DFS(level+1);
                    flag=1;
                }
                else if(i==4 && (recStack[level-1][3] + recStack[level-1][5] + recStack[level -1][2]<2*recStack[level-1][4]))
                {
                    funcStack[level]=4;
                    for(int j=1;j<=n;j++)
                    {
                        if(j!=4) recStack[level][j]=recStack[level-1][j];
                        else recStack[level][j]= recStack[level-1][3] + recStack[level-1][5] + recStack[level -1][2] - recStack[level-1][j];
                    }
                    DFS(level+1);
                    flag=1;
                }
                else if(i==3 && (recStack[level-1][1] + recStack[level-1][4]<2*recStack[level-1][3]))
                {
                    funcStack[level]=3;
                    for(int j=1;j<=n;j++)
                    {
                        if(j!=3) recStack[level][j]=recStack[level-1][j];
                        else recStack[level][j]= recStack[level-1][1] + recStack[level-1][4]- recStack[level-1][j];
                    }
                    DFS(level+1);
                    flag=1;
                }
            }
        }
    }
    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++;
```

```
        }
    }
}


int main()
{
    /*
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        getline(cin,S[i]);
        while(S[i].length()==0)
            getline(cin,S[i]);
    }
    */

    int i=1;
    while(getline(cin,T[i]))
    {
          getline(cin,S[i]);
//          while(S[i].length()==0)
//            getline(cin,S[i]);
        i++;
    }

    n=i-1;
    /*
    for(int i=1;i<=n;i++)
        cout<<S[i]<<"\n\n";
    */


    for(int i=1;i<=n;i++)
    {
        Minimal[i]=1;
        for(int j=1;j<=n;j++)
        {
            if(j==i)
                continue;
            if(isSubSequence(S[j],S[i]))
            {
                Minimal[i]=0;
                break;
            }
        }
    }


    cout<<"The Minimal Words are:\n";
    for(int i=1;i<=n;i++)
    {
        if(Minimal[i]==1)
            cout<<T[i]<<"\n"<<S[i]<<"\n";
    }


}
```