/\*

首先不考虑不合法的情况，由于要字典序最小，序列第一位自然应为A，易知所有大于0的系数所对应的位置都为A

系数所代表的是从原序列的第x位开始的子序列与原序列有长度为len的公共前缀，也意味着该序列的第len+1位与第x+len位不同，因此由n-1个有效数据可以得知原序列有n-1对元素不同，记录每一个元素与其前面的哪些元素不同就可以在O(n)的时间构造出原序列

接下来要判断构造出的序列是否合法，此时只需判断前缀是否相同，可以使用前缀和+hash暴力判断

\*/

#include<bits/stdc++.h>

using namespace std;

char c[5]={'X','A','C','G','T'};

int mod=10007;

int sum[100001]={0};

int poww[100001]={1,4};

int main()

{

freopen("dna.in","r",stdin);

freopen("dna.ans","w",stdout);

int T;

cin>>T;

while(T--)

{

int n;

cin>>n;

int d[100002];

int DNA[100002]={0};

vector<int> dif[100002];

bool impossible=false;

for(int i=1;i<=n;i++)

{

scanf("%d",&d[i]);

if(d[i]>n-i+1)

{

impossible=true;

}

if(i+d[i]<=n)dif[i+d[i]].push\_back(d[i]+1);

}

if(d[1]!=n)impossible=true;

if(impossible)

{

cout<<"Impossible\n";

continue;

}

for(int i=1;i<=n;i++)

{

bool available[5]={1,1,1,1,1};

for(int j=0;j<dif[i].size();j++)

{

available[DNA[dif[i][j]]]=false;

}

for(int a=1;a<=4;a++)

{

if(available[a])

{

DNA[i]=a;

break;

}

}

if(DNA[i]==0||(DNA[i]!=1&&d[i]>0))

{

impossible=true;

break;

}

}

if(impossible)

{

cout<<"Impossible\n";

continue;

}

for(int i=2;i<=n;i++)poww[i]=(poww[i-1]<<2)%mod;

for(int i=1;i<=n;i++)sum[i]=((sum[i-1]<<2)+DNA[i]-1)%mod;

for(int i=2;i<=n;i++)

{

if(d[i]==0)continue;

int temp=(sum[i+d[i]-1]-((sum[i-1]\*poww[d[i]])%mod)+mod)%mod;

if(temp!=sum[d[i]])

{

cout<<temp<<" "<<sum[d[i]]<<endl;

impossible=true;

break;

}

}

if(impossible)

{

cout<<"Impossible\n";

continue;

}

for(int i=1;i<=n;i++)printf("%c",c[DNA[i]]);

cout<<endl;

}

}