#include<bits/stdc++.h>

// implementataion of union by rank and path compression using trees

int rank[1000008];

int parent[100009];

using std::cout;

void makeset(int a)

{

parent[a]=a;

rank[a]=0;

}

int find(int a)

{

if(parent[a]==a)

return(a);

else

{parent[a]=find(parent[a]);//path compression

return(parent[a]);}

}

void union1(int a,int b)

{

parent[a]=find(a);

parent[b]=find(b);

if(parent[a]==parent[b])

return;

if(rank[parent[a]]>=rank[parent[b]])

{

if(rank[parent[a]]==rank[parent[b]])

rank[parent[a]]++;

parent[parent[b]]=parent[a];

}

else

{parent[parent[a]]=parent[b];}

}

int main()

{

makeset(1);

makeset(2);

makeset(3);

makeset(4);

makeset(5);

makeset(6);

makeset(7);

union1(1, 2);

union1(2, 3);

union1(4, 5);

union1(6, 7);

union1(5, 6);

// cout<<parent[7]<<"\*";

union1(3, 7);

//cout<<parent[7];

for(int i=1;i<=7;i++)

cout<<find(i)<<" ";

}