```
package adsprac3updated;
class Hashing
{
  int[] keys={20,50,53,75,100,67,105,3,36,39,6,23,29,56,87,99,35,79,83,41,71,12,8,60,80,112,132,108,115};
  int[] h1Arr, h2Arr;
  int count;
  boolean success,placed;
  public void replace(int k,int i,int e,int x)
  {
     int h1,h2,store;
     h1=e%x;
     h2=(e/x)%x;
     switch(i){
        case 1:
           if(h1Arr[h1]==-1)
           {
              h1Arr[h1]=e;
              placed=true;
              count=0;
              if(k==(keys.length-1))
              {
                 success=true;
                 return;
              }
              return;
           }
           else
           {
              store=h1Arr[h1];
              h1Arr[h1]=e;
              count++;
              if(count<k)
              {
                 replace(k,2,store,x);
                 if(k==(keys.length-1))
                 {
                   success=true;
                   return;
                 }
              }
              else
              {
                 success=false;
                 System.out.println("Cycle found");
                 return;
              }
           }
           break;
        case 2:
           if(h2Arr[h2]==-1)
           {
              h2Arr[h2]=e;
              placed=true;
```

```
            count=0;
            if(k==(keys.length-1))
            {
               success=true;
               return;
            }
          }
          else
          {
            store=h2Arr[h2];
            h2Arr[h2]=e;
            count++;
            if(count<k)
            {
               replace(k,1,store,x);
               if(k==(keys.length-1))
               {
                 success=true;
                 return;
               }
            }
            else
            {
               success=false;
               System.out.println("Cycle found");
               return;
            }
          }
          break;
       default: System.out.println("Not possible\n");
    }
}
public boolean hash(int x1)
{
   System.out.println(keys.length+"  "+x1);
   h1Arr=new int[x1];
   h2Arr=new int[x1];
   for(int i=0;i<x1;i++)
    {
      h1Arr[i]=-1;
      h2Arr[i]=-1;
    }
   for(int i=0;i<keys.length;i++)
   {
      placed=false;
      int h1,store;
      h1=keys[i]% x1;
        if(h1Arr[h1]==-1)
        {
          h1Arr[h1]=keys[i];
          //System.out.println("element "+keys[i]+" inserted in 1 at location "+h1);
          count=0;
          if(i==(keys.length-1))
```

```
            {
               success=true;
            }
           }
          else
          {
             store=h1Arr[h1];
             h1Arr[h1]=keys[i];
             //System.out.println("element "+keys[i]+" inserted in 1 at location "+h1);
             count++;
             if(count<=i)
             {
                replace(i,2,store,x1);
                if(placed==true)
                {
                if(i==(keys.length-1))
                {
                   success=true;
                   break;
                }
                }
                else
                {
                   success=false;
                   break;
                }
             }
             else
             {
                success=false;
                System.out.println("Cycle found");
                break;
             }
          }
     }
     System.out.println("\nH1: ");
     for(int j=0;j<h1Arr.length;j++)
     {
        System.out.print(h1Arr[j]+"  ");
     }
     System.out.println("\nH2: ");
     for(int j=0;j<h2Arr.length;j++)
     {
        System.out.print(h2Arr[j]+"  ");
     }
  return success;
}
}

public class ADSPrac3Updated {
   public static void main(String[] args) {
      int[] mod={11,13,15,17};
      Hashing hf1=new Hashing();
```

```
    for(int p=0;p<mod.length;p++)
    {
        boolean v = hf1.hash(mod[p]);
        System.out.println(v);
          if(v==true)
          {
             System.out.println("Hasing Completed and no cycle found\n");
             break;
          }
          else
          {
             if(p==3)
                System.out.println("Not possible");
             else
                System.out.println("\nCycle found. So increasing the table size\n ");
          }
    }
  }
}
```

OUTPUT:

```
29  11
Cycle found
H1:
-1  67  -1  3  -1  -1  105  -1  -1  53  -1
H2:
6  20  -1  36  50  -1  75  -1  -1  100  -1  false
Cycle found. So increasing the table size

29  13
Cycle found
H1:
39  79  67  3  56  83  71  20  8  35  36  50  12
H2:
6  23  29  41  53  75  87  99  105  -1  -1  -1  -1  false
Cycle found. So increasing the table size

29  15
Cycle found
H1:
75  -1  -1  3  79  20  36  67  53  39  100  41  87  -1  29
H2:
6  23  35  56  -1  83  99  105  -1  -1  -1  -1  -1  -1  -1  false
Cycle found. So increasing the table size

29  17
Cycle found
H1:
-1  -1  36  3  -1  56  23  75  -1  -1  -1  -1  29  -1  -1  100  67
H2:
6  20  50  53  -1  -1  105  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  false
Not possible
```