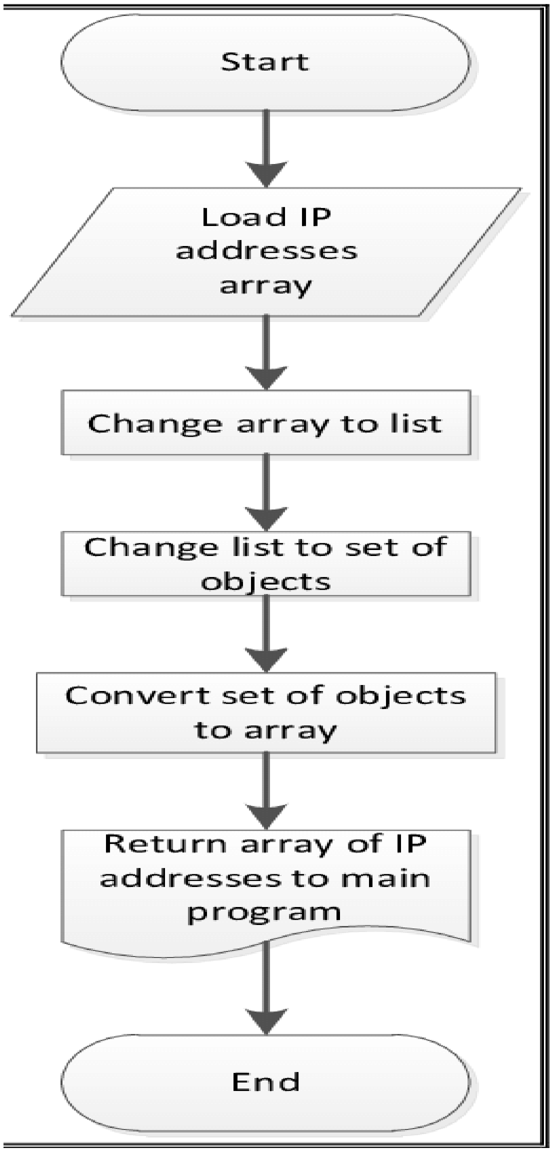
```java
public class RemoveDuplicate {

    //Represent a node of the singly linked list
    class Node{
        int data;
        Node next;

        public Node(int data) {
            this.data = data;
            this.next = null;
        }
    }

    //Represent the head and tail of the singly linked list
    public Node head = null;
    public Node tail = null;

    //addNode() will add a new node to the list
    public void addNode(int data) {
        //Create a new node
        Node newNode = new Node(data);

        //Checks if the list is empty
        if(head == null) {
            //If list is empty, both head and tail will point to new node
            head = newNode;
            tail = newNode;
        }
        else {
            //newNode will be added after tail such that tail's next will point to newNode
            tail.next = newNode;
            //newNode will become new tail of the list
            tail = newNode;
        }
    }

    //removeDuplicate() will remove duplicate nodes from the list
    public void removeDuplicate() {
        //Node current will point to head
        Node current = head, index = null, temp = null;

        if(head == null) {
            return;
        }
        else {
            while(current != null){
                //Node temp will point to previous node to index.
                temp = current;
                //Index will point to node next to current
                index = current.next;

                while(index != null) {
                    //If current node's data is equal to index node's data
                    if(current.data == index.data) {
                        //Here, index node is pointing to the node which is duplicate of current node
                        //Skips the duplicate node by pointing to next node
                        temp.next = index.next;
                    }
                    else {
                        //Temp will point to previous node of index.
                        temp = index;
                    }
                    index = index.next;
                }
                current = current.next;
            }
        }
    }

    //display() will display all the nodes present in the list
    public void display() {
        //Node current will point to head
        Node current = head;
        if(head == null) {
            System.out.println("List is empty");
            return;
        }
        while(current != null) {
            //Prints each node by incrementing pointer
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {

        RemoveDuplicate sList = new RemoveDuplicate();

        //Adds data to the list
        sList.addNode(1);
        sList.addNode(2);
        sList.addNode(3);
        sList.addNode(2);
        sList.addNode(2);
        sList.addNode(4);
        sList.addNode(1);

        System.out.println("Originals list: ");
                    sList.display();

        //Removes duplicate nodes
        sList.removeDuplicate();

        System.out.println("List after removing duplicates: ");
        sList.display();
    }
}
```

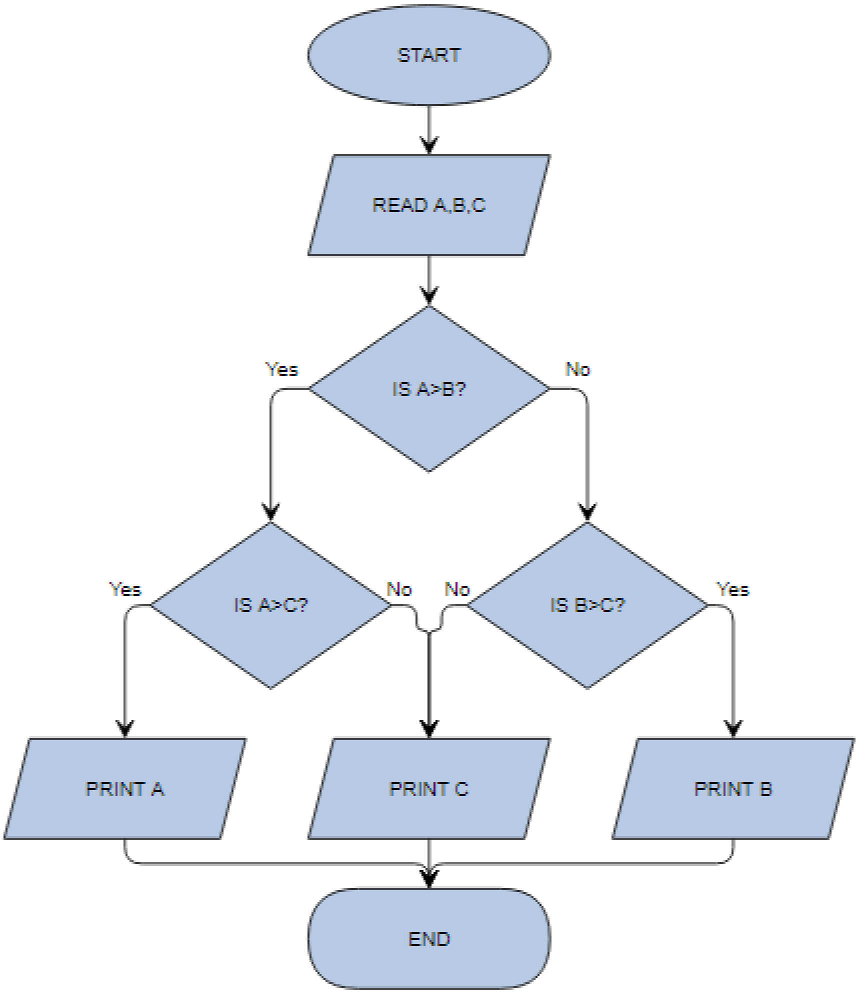

Start
Load IP addresses array
Change array to list
Change list to set of objects
Convert set of objects to array
Return array of IP addresses to main program
End

```
class NumberToWord
```

```
{
//user-defined static method that converts a number into words
static void numberToWords(char num[])
{
//determines the number of digits in the given number
int len = num.length;
//checks the given number has number or not
if (len == 0)
{
//if the given number is empty prints the following statement
System.out.println("The string is empty.");
return;
}
//here, we have specified the length of the number to 4
//it means that the number (that you want to convert) should be four or less than four digits
if (len > 4)
{
//if the given number is more than four-digit number, it prints the following statement
System.out.println("\n The given number has more than 4 digits.");
return;
}
//string type array for one-digit numbers
String[] onedigit = new String[] {"Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};
//string type array for two digits numbers
//the first index is empty because it makes indexing easy
String[] twodigits = new String[] {"", "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};
//string type array of tens multiples
//the first two indexes are empty because it makes indexing easy
String[] multipleoftens = new String[] {"",  "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};
```

```
//string type array of power of tens
String[] poweroftens = new String[] {"Hundred", "Thousand"};
//Used for debugging purpose only
//the valueOf() method returns the string representation of the character array argument
System.out.print(String.valueOf(num) + ": ");
//checks whether the length of the given string is one or not
if (len == 1)
{
//if the above condition returns true, it accesses the corresponding index and prints the value of that index
//[num[0]-'0']: getting the number equal the decimal value of the character (assuming the char is the digit)
System.out.println(onedigit[num[0]-'0']);
return;
}
int x = 0;
//executes until num does not become not '\0'
while (x < num.length)
{
//executes if the length of the string is greater than equal to three
if (len >= 3)
{
if (num[x] - '0' != 0)
{
System.out.print(onedigit[num[x] - '0'] + " ");
//here length can be 3 or 4
System.out.print(poweroftens[len - 3]+ " ");
}
//decrements the length of the string by 1
--len;
}
//executes if the given number has two digits
```

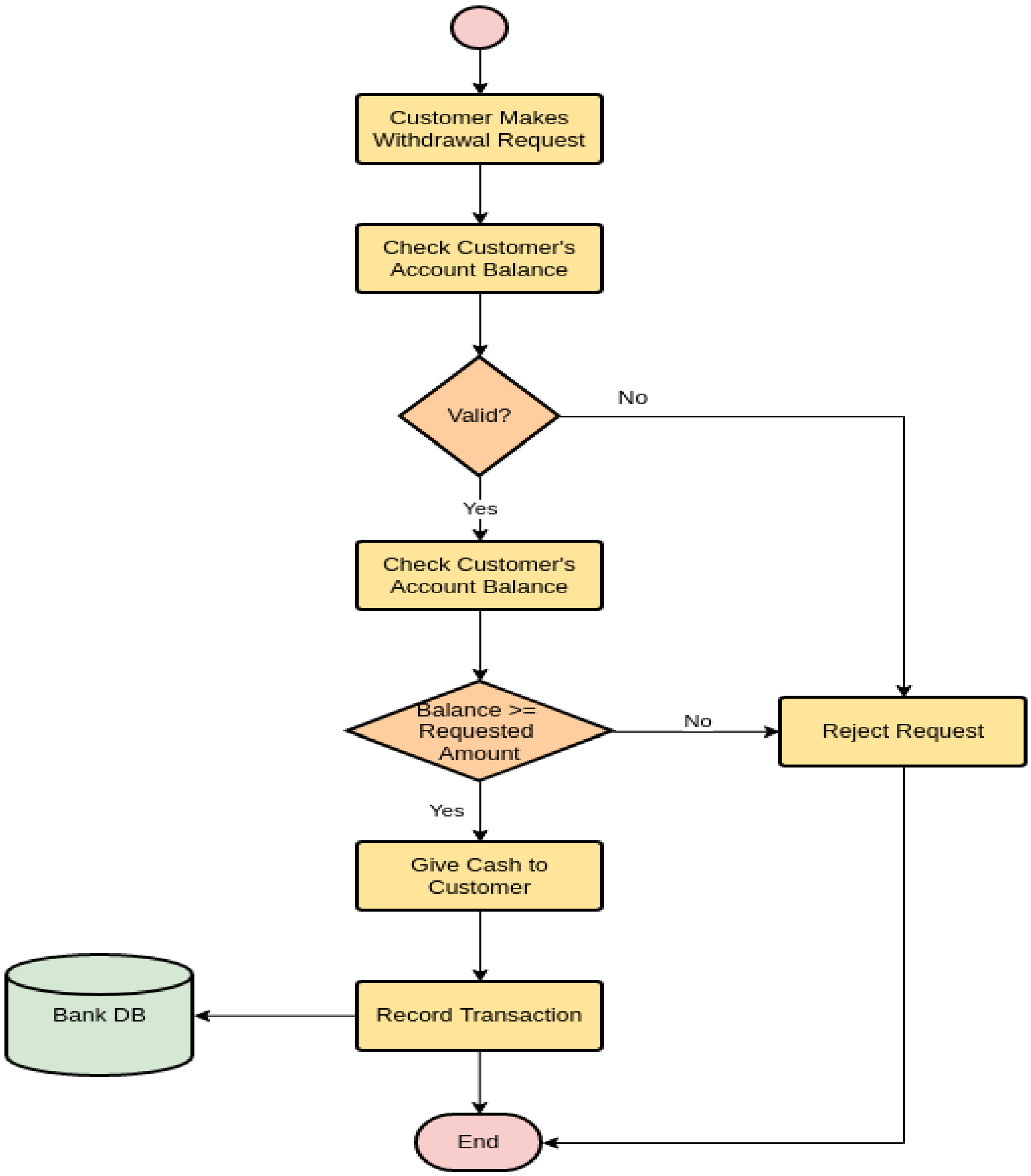
```
else
{
//the if-statement handles the numbers from 10 to 19 only
if (num[x] - '0' == 1)
{
//adding the digits of the given number
//the logic behind sum up the digits is that we will use the sum for accessing the index of the array
//for example: 17, sum of digits = 8
//we will access the 8th index in twodigits[] array i.e. Seventeen
int sum = num[x] - '0' + num[x + 1] - '0';
System.out.println(twodigits[sum]);
return;
}
//the else-if statement handles the number 20 only
//compares the tens and unit place with 2 and 0 respectively
else if (num[x] - '0' == 2 && num[x + 1] - '0' == 0)
{
//executes if the above else-if condition returns true
System.out.println("Twenty");
return;
}
//the else block handles the numbers from 21 to 100
else
{
int i = (num[x] - '0');
if (i > 0)
//prints the ith index element of the array multipleoftens[]
System.out.print(multipleoftens[i]+ " ");
else
//prints space
System.out.print("");
//increments the variable i by 1
++x;
//checks whether the number is not equal to zero, it means the number has only a digit
if (num[x] - '0' != 0)
//prints the ith index element of the array onedigit[]
System.out.println(onedigit[num[x] - '0']);
}
}
//increments the variable i by 1
++x;
}
}
//main() method
public static void main(String args[])
{
//calling the user-defined method and that invokes another predefined method toCharArray()
//the method toCharArray() converts the given number into character array
numberToWords("1111".toCharArray());
numberToWords("673".toCharArray());
numberToWords("85".toCharArray());
numberToWords("5".toCharArray());
numberToWords("0".toCharArray());
numberToWords("20".toCharArray());
numberToWords("1000".toCharArray());
numberToWords("12345".toCharArray());
//passing empty string
numberToWords("".toCharArray());
}
}
```



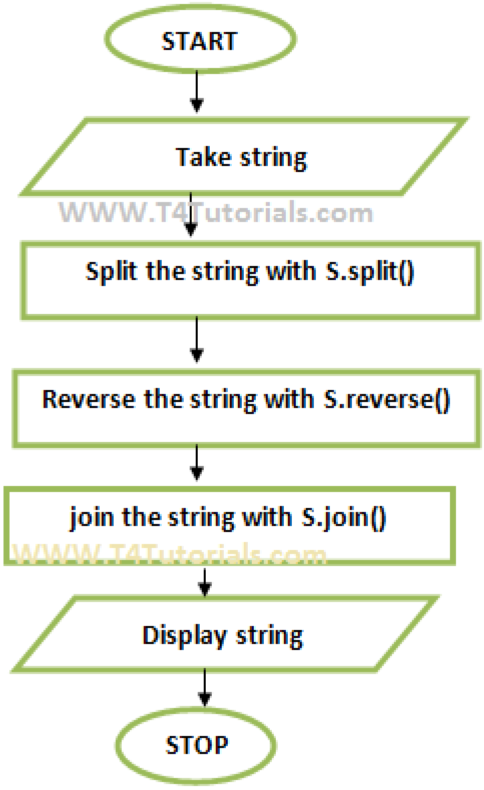
```java
//import required classes and packages
import java.util.Scanner;

//create ATMExample class to implement the ATM functionality
public class ATMExample
{
    //main method starts
    public static void main(String args[] )
    {
        //declare and initialize balance, withdraw, and deposit
        int balance = 100000, withdraw, deposit;

```

```
//create scanner class object to get choice of user
Scanner sc = new Scanner(System.in);

while(true)
{
    System.out.println("Automated Teller Machine");
    System.out.println("Choose 1 for Withdraw");
    System.out.println("Choose 2 for Deposit");
    System.out.println("Choose 3 for Check Balance");
    System.out.println("Choose 4 for EXIT");
    System.out.print("Choose the operation you want to perform:");

    //get choice from user
    int choice = sc.nextInt();
    switch(choice)
    {
        case 1:
System.out.print("Enter money to be withdrawn:");

//get the withdrawl money from user
withdraw = sc.nextInt();

//check whether the balance is greater than or equal to the withdrawal amount
if(balance >= withdraw)
{
    //remove the withdrawl amount from the total balance
    balance = balance - withdraw;
    System.out.println("Please collect your money");
}
else
{
    //show custom error message
    System.out.println("Insufficient Balance");
}
```

```
    System.out.println("");
    break;

        case 2:

    System.out.print("Enter money to be deposited:");

    //get deposite amount from te user
    deposit = sc.nextInt();

    //add the deposit amount to the total balanace
    balance = balance + deposit;
    System.out.println("Your Money has been successfully depsited");
    System.out.println("");
    break;

        case 3:
    //displaying the total balance of the user
    System.out.println("Balance : "+balance);
    System.out.println("");
    break;

        case 4:
    //exit from the menu
    System.exit(0);
      }
    }
}
}
```


Customer Makes Withdrawal Request
Check Customer's Account Balance
Valid?
No
Yes
Check Customer's Account Balance
Balance >= Requested Amount
No
Reject Request
Yes
Give Cash to Customer
Bank DB
Record Transaction
End

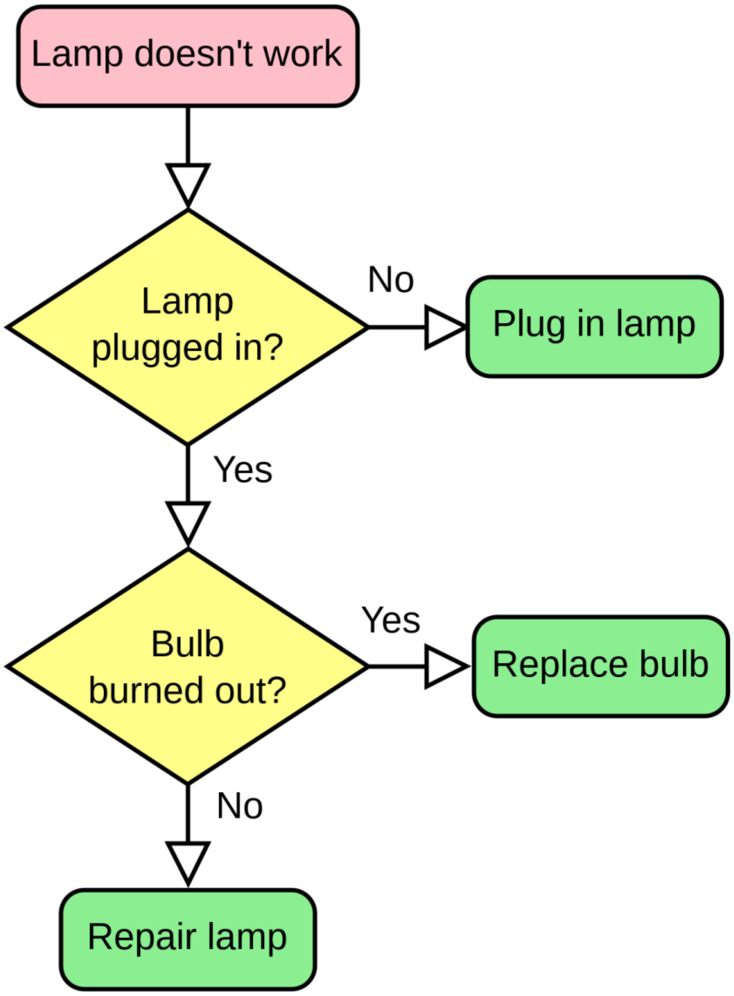
```
import java.util.Scanner;
public class ReverseStringExample1
{
public static void main(String[] args)
{
String str;
System.out.println("Enter a string: ");
Scanner scanner = new Scanner(System.in);
str = scanner.nextLine();
scanner.close();                                   //closes the input stream
String reversed = reverseString(str);
System.out.println("The reversed string is: " + reversed);
}
public static String reverseString(String s)
{
if (s.isEmpty())                    //checks the string if empty
return s;
return reverseString(s.substring(1)) + s.charAt(0);            //recursively called function
}
}
```


START
Take string
WWW.T4Tutorials.com
Split the string with S.split()
Reverse the string with S.reverse()
join the string with S.join()
WWW.T4Tutorials.com
Display string
STOP

```
public class RotateList {

    //Represent a node of the doubly linked list

    class Node{
        int data;
        Node previous;
        Node next;

        public Node(int data) {
            this.data = data;
        }
    }

    int size = 0;
    //Represent the head and tail of the doubly linked list
    Node head, tail = null;

    //addNode() will add a node to the list
    public void addNode(int data) {
        //Create a new node
        Node newNode = new Node(data);

        //If list is empty
        if(head == null) {
            //Both head and tail will point to newNode
            head = tail = newNode;
            //head's previous will point to null
            head.previous = null;
            //tail's next will point to null, as it is the last node of the list
            tail.next = null;
        }
        else {
            //newNode will be added after tail such that tail's next will point to newNode
            tail.next = newNode;
            //newNode's previous will point to tail
            newNode.previous = tail;
            //newNode will become new tail
            tail = newNode;
            //As it is last node, tail's next will point to null
            tail.next = null;
        }
        //Size will count the number of nodes present in the list
        size++;
    }

    //rotateList() will rotate the list by given n nodes
    public void rotateList(int n) {
        //Initially, current will point to head
        Node current = head;

        //n should not be 0 or greater than or equal to number of nodes present in the list

        if(n == 0 || n >= size)
            return;
        else {
            //Traverse through the list till current point to nth node
            //after this loop, current will point to nth node
            for(int i = 1; i < n; i++)
                current = current.next;

            //Now to move entire list from head to nth node and add it after tail
            tail.next = head;
            //Node next to nth node will be new head
            head = current.next;
            //Previous node to head should be null
            head.previous = null;
            //nth node will become new tail of the list
            tail = current;
            //tail's next will point to null
            tail.next = null;
        }
    }

```

```
//display() will print out the nodes of the list
public void display() {
    //Node current will point to head
    Node current = head;
    if(head == null) {
        System.out.println("List is empty");
        return;
    }
    while(current != null) {
        //Prints each node by incrementing the pointer.

        System.out.print(current.data + " ");
        current = current.next;
    }
    System.out.println();
}

public static void main(String[] args) {

    RotateList dList = new RotateList();
    //Add nodes to the list
    dList.addNode(1);
    dList.addNode(2);
    dList.addNode(3);
    dList.addNode(4);
    dList.addNode(5);

               System.out.println("Original List: ");
               dList.display();

               //Rotates list by 3 nodes
               dList.rotateList(3);

               System.out.println("Updated List: ");
```

```
            dList.display();
        }
    }
```



Lamp doesn't work
Lamp plugged in?
No
Plug in lamp
Yes
Bulb burned out?
Yes
Replace bulb
No
Repair lamp