1. Phone Number Checking Problem:

**Program Title:**

Write a program which checks valid phone numbers from a set of data from file.

**Objective:**

To learn phone number checking system using **Finite Automata**.

**Theory:**

The following program can be solved by the corresponding **DFA**:

+ 8 8 0 1 [3, 5-9] 0-9

0-9

0-9 0-9 0-9 0-9 0-9 0-9

1. Checks number length is 14 or not.

2. Checks country code.

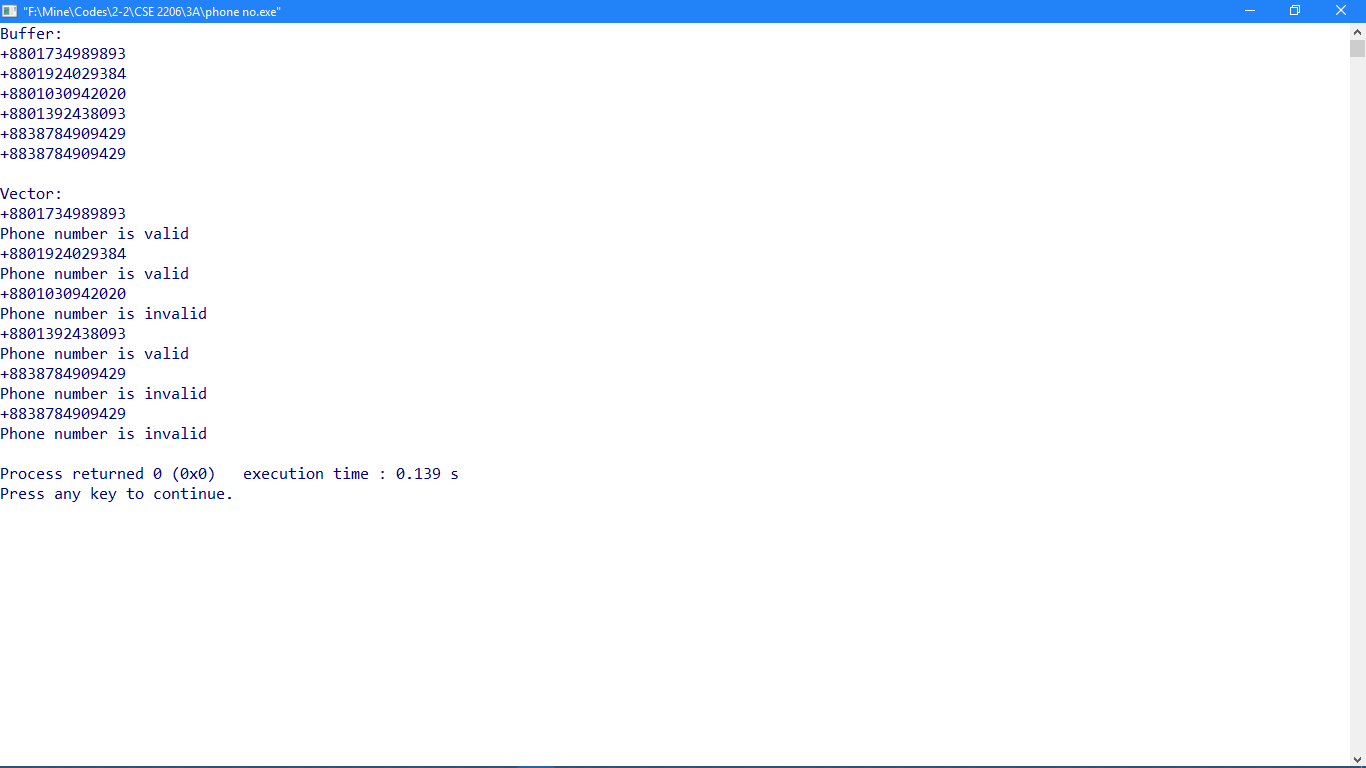
3. Checks network operator.

4. Checks other digits are correct or not.

**Source Code:**

1. #include <iostream>
2. #include <fstream>
3. #include <vector>
4. **using** **namespace** std;
5. string buffer, temp;
6. vector<string>phoneNo;
7. **int** phoneNoAutomata (string phoneNumber);
9. **int** main()
10. {
11. ifstream f1;
12. f1.open("phone no.txt");
13. cout << "Buffer:\n";
14. **while**(!f1.eof()){
15. f1 >> buffer;
16. phoneNo.push\_back(buffer);
17. cout << buffer << endl;
18. }
19. cout << "\nVector:\n";
20. **for**(**int** i = 0; i < phoneNo.size(); i ++){
21. cout << phoneNo[i] << endl;
22. phoneNoAutomata(phoneNo[i]);
23. }
24. }
25. **int** phoneNoAutomata (string phoneNumber)
26. {
27. string phoneNumberFirstPart = phoneNumber.substr(0, 5);
28. **int** flag = 1;
29. **for**(**int** i = 1; i < phoneNumber.size(); i++){
30. **if**(!isdigit(phoneNumber[i]))
31. flag = 0;
32. }
33. **if**(phoneNumber.size() != 14)
34. flag = 0;
35. **if**(phoneNumberFirstPart != "+8801")
36. flag = 0;
37. **if**(phoneNumber[5] == '0' || phoneNumber[5] == '1' || phoneNumber[5] == '2' || phoneNumber[5] == '4')
38. flag = 0;
39. **if**(flag == 0)
40. cout << "Phone number is invalid\n";
41. **else**
42. cout << "Phone number is valid\n";
43. }

**Output:**



2. OTP generation:

**Program Title:**

Write a program to generate some **OTP** of 6 digits like “RUET –XXXXXX”.

**Objective:**

To learn to generate **OTP**.

**Theory:**

A one-time password (**OTP**) is an automatically generated numeric or alphanumeric string of characters that authenticates the user for a single transaction or login session. Here **String catenation** is used for “RUET- “substring and **Modular Congruence** method for generating random OTPs.

”

”

**Source Code:**

1. #include <iostream>
2. #include <fstream>
4. **using** **namespace** std;
5. **int** main()
6. {
7. ofstream fout;
8. fout.open("OTP.txt");
9. **int** a, c, count = 0;
10. **long** **int** m, x0, xN;
11. a = 1999, c = 3, m = 10000000, x0 = 7;
12. **for**(**int** i = 0; i < 1000; i++){
13. xN = ((a \* x0) + c) % m;
14. **if**(xN > 999999 && xN < 9999999){
15. count++;
16. fout << "RUET-" <<  xN << "\n";
17. cout << "RUET-" << xN <<  "\n";
18. }
19. x0 = xN;
20. }
21. cout << "Total OTPs: " << count << endl;
22. }

**Output:**

