I used C++ programming language to write the program. Because I have more knowledge on this language than I have on Java also I think C++ is more flexible.

First of all I read the NFA file with fstream library. In a while loop, I assigned the data to vectors.

```
if(readingState.compare("ALPHABET")==0){
    alphabetInput.push_back(myText);
```

readingState variable in here helps to read header values of data and to assign data to correct position. If we come across new header such as “STATES” readingState’s value setted to “STATES” and state values are assigned to correct vector.

```
else if(readingState.compare("STATES")==0){
    stateInput.push_back(myText);
```

After we read the data. We start to create NFA class objects as an array because we may not know how many state there will be for every program, we decide it after we read the data.

```
class NfaState{
public:

string Name;
string Alphabet;
vector<char> Transitions;


};
```

```
NfaState state[stateLength];
```

Now that we created right amount of objects we can set their attributes. First we assign the States as names to our objects.

```
index = 0;
for (auto i = stateInput.begin(); i != stateInput.end(); i++){
    state[index].Name =*i;
    cout<<(state[index].Name)<<endl;
    index++;
}
```

We have an extra index value here because in for loop we use vector queue as index. After this we can assign transitions of every state to their objects.

```
for(auto i = transitionInput.begin(); i != transitionInput.end(); i++){
    string val = *i;
    for(int k = 0; k<stateLength; k++){
            if(val[0] == state[k].Name[0]){
                state[k].Transitions.push_back(val[4]);
                state[k].Transitions.push_back(val[2]);
            }
    }
}
```

In order to assign the transitions to correct state we use an if condition which is basically assigns the values if char at 0 which is the name of the state is equals to state name. It assigns the next state and the alphabet used in order to go next state. Output is shown below.

```
A A0
A A1
A B1
B C1
```

The first value is name of the state and the second values are in order of next state and the alphabet used in order to go this state. Right values are hold in transaction attribute of the object. For every Transaction attribute of the object the even index numbers are next state and the odd index numbers in the vector are alphabet used.

Now we will create DFA class which it's attributes are the same with NFA class.

```
class DfaState{
    public:
    string Name;
    string Alphabet;
    vector<char>Transitions;
};
```

As we know in conversion of NFA to DFA we combine two states if there are multiple options to change state and we create new state from this combination. In program we search through the Transitions attribute if there are multiple choices of changing state, we combine these two state and create new state such as {AB}.

After that, for the new state we created we unite the transactions of the two NFA state that we used to create this new state and assign to its transitions attributes.

```
for (int i = 0; i< stateLength; i++){
    nameCounter1 = 0;
    nameCounter2 = 0;
    for( auto j = state[i].Transitions.begin(); j != state[i].Transitions.end(); j++){
        if(state[i].Name[0] == *j){
            nameCounter1++;
        }
        else if (*j != state[i].Alphabet[0] && *j != state[i].Alphabet[1] ){
            nameCounter2++;
        }
        if(nameCounter1>=2 && nameCounter2>0){
            DfaState dfa;
            dfa.Name = state[i].Name + "1";
            dfas.push_back(dfa);

        }
    }

}
```