main.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;
string round_keys[16];
string shift_left_once(string key_chunk){
    string shifted="";
        for(int i = 1; i < 28; i++){
            shifted += key_chunk[i];
        }
        shifted += key_chunk[0];
    return shifted;
}
string shift_left_twice(string key_chunk){
    string shifted="";
    for(int i = 0; i < 2; i++){
        for(int j = 1; j < 28; j++){
            shifted += key_chunk[j];
        }
        shifted += key_chunk[0];
        key_chunk= shifted;
        shifted ="";
    }
    return key_chunk;
}
void generate_keys(string key){
    int pc1[56] = {
    57,49,41,33,25,17,9,
    1,58,50,42,34,26,18,
    10,2,59,51,43,35,27,
    19,11,3,60,52,44,36,
    };
    int pc2[48] = {
    14,17,11,24,1,5,
    3,28,15,6,21,10,
    23,19,12,4,26,8,
    16,7,27,20,13,2,
    41,52,31,37,47,55,
    30,40,51,45,33,48,
    44,49,39,56,34,53,
    46,42,50,36,29,32
    };
    // 1. Compressing the key using the PC1 table
    string perm_key ="";
    for(int i = 0; i < 56; i++){
        perm_key+= key[pc1[i]-1];
    }
    // 2. Dividing the result into two equal halves
    string left= perm_key.substr(0, 28);
    string right= perm_key.substr(28, 28);
    // Generating 16 keys
    for(int i=0; i<16; i++){
        if(i == 0 || i == 1 || i==8 || i==15 ){
            left= shift_left_once(left);
            right= shift_left_once(right);
        }
        else{
            left= shift_left_twice(left);
            right= shift_left_twice(right);
        }
```

```
    // 2. Dividing the result into two equal halves
    string left= perm_key.substr(0, 28);
    string right= perm_key.substr(28, 28);
    // Generating 16 keys
    for(int i=0; i<16; i++){
        if(i == 0 || i == 1 || i==8 || i==15 ){
            left= shift_left_once(left);
            right= shift_left_once(right);
        }
        else{
            left= shift_left_twice(left);
            right= shift_left_twice(right);
        }
    string combined_key = left + right;
    string round_key = "";
    for(int i = 0; i < 48; i++){
        round_key += combined_key[pc2[i]-1];
    }
    round_keys[i] = round_key;
        cout<<"Key "<<i+1<<": "<<round_keys[i]<<endl;
    }
}
int main(){
    string key = "1010101010111011000010010001100000100111001101101100110011011101"
    "01101100110011011101";
    cout<<"input key: "<<key<<endl;
    cout<<"Genrated keys"<<endl;
    generate_keys(key);
}
```

OUTPUT:

```
input key: 1010101010111011000010010001100000100111001101101100110011011101
Genrated keys
Key 1: 000110010100110011010000011100101101111010001100
Key 2: 010001010110100001011000000110101011110011001110
Key 3: 000001101110110110100100101011001111010110110101
Key 4: 110110100010110100000011001010110110110111011100011
Key 5: 011010011010011000101001111111110110010010010011
Key 6: 110000011001010010001110100001110100011101011110
Key 7: 011100001000101011010010110111011011001111000000
Key 8: 001101001111100000100010111100001100011001101101
Key 9: 100001001011101101000100011100111101110011001100
Key 10: 000000100111011001010111000010001011010110111111
Key 11: 011011010101010101100000101011110111110010100101
Key 12: 110000101100000111101001011010100100101111110011
Key 13: 100110011100001100010011100101111100100100011111
Key 14: 001001010001101110001011110001110001011111010000
Key 15: 001100110011000011000101110110011010001101101101
Key 16: 000110000001110001011101011101011100011001101101
```