```cpp
#include <iostream>
#include <string>
#include <vector>
#include <map>

int main()
{
    std::string key;
    std::string plain;

    //Create a map to hold all of the valid characters to be used (A-Z, 1-9, '
     ')
    std::map<char, int>valid_characters;

    //Inserts all valid characters into the map
    for (int i = 32; i <= 90; i++)
    {
        if ((i > 32 && i <= 48) || (i > 57 && i <= 64))
        {
            continue;
        }
        valid_characters.insert(std::pair<char,int>(char(i), 0));
    }

    //Gets key for ciphertable
    std::cout << "Please input your key: ";
    std::getline(std::cin, key);

    //Checks if the inputted key is valid with usable characters
    for (int i = 0; i < key.size(); i++)
    {
        //Changes '0' to 'O' in the key string to be put into the table
        if (key[i] == '0')
        {
            key[i] = 'O';
        }
        if(valid_characters.find(key[i]) == valid_characters.end())
        {
            std::cout << "Please provide a valid key!" << std::endl;
            return 0;
        }
    }

    //Gets the string to be encoded
    std::cout << "Please input string to be encoded: ";
    std::getline(std::cin, plain);

    //Checks if the inputted string is valid with usable characters
    for (int i = 0; i < plain.size(); i++)
    {
        //Changes '0' to 'O' in the plain string to be encrypted
        if (plain[i] == '0')
        {
            plain[i] = 'O';
        }
        if(valid_characters.find(plain[i]) == valid_characters.end())
        {
            std::cout << "Please provide a valid string to be encoded!" <<
             std::endl;
            return 0;
        }
    }

    //Create 6x6 vector to hold the ciphertable
    std::vector<std::vector<char> > cipher_table(6, std::vector<char>(6, '#'));

    int k = 0;
    //Sets up the cipher table with the key
    for (int i = 0; i < 6; i++)
    {
        for (int j = 0; j < 6; j++)
        {
            // Checks to see if the character is already in the table and if
             so go to the next character in the key string
            while(valid_characters[key[k]] == 1 && k < key.size())
            {
                k++;
            }
            if (k < key.size())
            {
                cipher_table[i][j] = key[k];
                valid_characters[key[k]] = 1;
            }
            else
            {
                break;
            }
        }
        if(k >= key.size())
        {
            break;
        }
    }

    //Put A-Z in ciphertable
    k = 65;
    for (int i = 0; i < 6; i++)
    {
        for (int j = 0; j < 6; j++)
        {
            // Checks to see if the character is already in the table and if
             so go to the next valid character
        while (valid_characters[char(k)] == 1)
        {
            k++;
            if (k == 91)
            {
                break;
            }
        }
        if (k == 91)
        {
            break;
        }

        // Checks to see if the cell we are trying to use for the
         character is not used yet
        if (cipher_table[i][j] == '#')
        {
            cipher_table[i][j] = char(k);
            valid_characters[char(k)] = 1;
        }
    }
    if (k == 91)
    {
        break;
    }
}

//Put 1-9 into ciphertable
k = 49;
for (int i = 0; i < 6; i++)
{
    for (int j = 0; j < 6; j++)
    {
        // Checks to see if the character is already in the table and if
         so go to the next valid character
        while (valid_characters[char(k)] == 1)
        {
            k++;
            if (k == 58)
            {
                break;
            }
        }
        if (k == 58)
        {
            break;
        }

        // Checks to see if the cell we are trying to use for the
         character is not used yet
        if (cipher_table[i][j] == '#')
        {
            cipher_table[i][j] = char(k);
            valid_characters[char(k)] = 1;
        }
    }
    if (k == 58)
    {
        break;
    }
}

//Put space into cipher table if its not already in it
if (cipher_table[5][5] == '#')
{
    cipher_table[5][5] = char(32);
    valid_characters[char(32)] = 1;
}


/* Check cipher table to see if it is right, comment out */
for(int i = 0; i < 6; i++)
{
    for (int j = 0; j < 6; j++)
    {
        std::cout << cipher_table[i][j] << " ";
    }
    std::cout << std::endl;
}

//Start the cipher with the cipher table and plain text

//Puts an 'X' between characters that are repeated in the plain text
for (int i = 0; i < plain.size() - 1; i += 2)
{
    if (plain[i] == plain[i + 1])
    {
        plain.insert(i + 1, "X");
    }
}

//Check if the plain string with the X's is an even number if not insert
 an 'X' at the end
if (plain.size() % 2 != 0)
{
    plain.push_back('X');
}


//Make a map that has the x,y cooridinates for characters in the
 cipher_table so that we know their coordinates for replacing them
std::map<char, std::pair<int, int> >table_grid;
```

```
for (int i = 0; i < 6; i++)
{
    for (int j = 0; j < 6; j++)
    {
        table_grid.insert(std::make_pair(cipher_table[i][j],
         std::make_pair(i, j)));
    }
}

//encrypt the plain text
std::string encrypted_text;
for (int i = 0; i < plain.size() - 1; i += 2)
{
    //Get the coordinates for 2 letters next to eachother
    int x_1 = table_grid[plain[i]].second;
    int y_1 = table_grid[plain[i]].first;
    int x_2 = table_grid[plain[i + 1]].second;
    int y_2 = table_grid[plain[i + 1]].first;

    //Check to see if the letters are in same row we replace with 1 letter
     to the right
    if (y_1 == y_2)
    {
        //If x_1 + 1 > 5 we know that we have to go to the first column
         otherwise we just move over 1 column
        if (x_1 + 1 > 5)
        {
            encrypted_text += cipher_table[y_1][0];
        }
        else
        {
            encrypted_text += cipher_table[y_1][x_1 + 1];
        }

        //If x_2 + 1 > 5 we know that we have to go to the first column
         otherwise we just move over 1 column
        if (x_2 + 1 > 5)
        {
            encrypted_text += cipher_table[y_2][0];
        }
        else
        {
            encrypted_text += cipher_table[y_2][x_2 + 1];
        }
    }
    // Check to see if the letters are in the same column then replace
     with 1 row down
    else if(x_1 == x_2)
    {
        //If y_1 + 1 > 5 we know that we have to go to the first row
         otherwise we just move down 1 row
```

```
            if (y_1 + 1 > 5)
            {
                encrypted_text += cipher_table[0][x_1];
            }
            else
            {
                encrypted_text += cipher_table[y_1 + 1][x_1];
            }

            // If y_2 + 1 > 5 we know that we have to go to the first row otherwise we just move down 1 row
            if (y_2 + 1 > 5)
            {
                encrypted_text += cipher_table[0][x_2];
            }
            else
            {
                encrypted_text += cipher_table[y_2 + 1][x_2];
            }
        }
        // If they are not in the same row or column we get the character from the same column as the other but in its original row
        else
        {
            encrypted_text += cipher_table[y_1][x_2];
            encrypted_text += cipher_table[y_2][x_1];
        }
    }

    //Output the encrypted message
    std::cout << encrypted_text << std::endl;

    return 0;
}
```