Program :

```
#include <iostream>
#include<cmath>
using namespace std;
int row(char A[5][5],char ch)
{
   int i,j;
   for(i=0;i<5;i++)
      for(j=0;j<5;j++)
         if(A[i][j]==ch)
            return i;
   return -1;
}
int column(char A[5][5],char ch)
{
   int i,j;
   for(i=0;i<5;i++)
      for(j=0;j<5;j++)
         if(A[i][j]==ch)
            return j;
   return -1;
}
void constructPlayFairMat(char playfair[5][5],string key)
{
   char ch='a';
   int i,j,k=0;
   for(i=0;i<5;i++)
   {
      for(j=0;j<5;j++)
      {

         while(k<key.length() && row(playfair,key[k])!=-1)
            k++;
         if(k==key.length()) break;
         playfair[i][j]=key[k];
         k++;
      }
      if(k==key.length()) break;
   }
   while(i<5)
   {

      while(j<5)
      {
         while(ch<122 && (row(playfair,ch)!=-1 || ch=='j'))
            ch++;
         playfair[i][j]=ch;
         ch++;
         j++;
      }
      j=0;
```

```
        i++;
    }
}
string constructDigrams(string temp)
{
    int i;
    for(i=0;i<ceil(temp.length()/2)*2;i+=2)
    {
        if(temp[i]==temp[i+1])
        {
            temp=temp.substr(0,i+1).append(1,'x').append(temp.substr(i+1,temp.length()-i-1));
        }
    }
    if(temp.length()%2!=0)
        temp.append(1,'x');
    return temp;
}
string encrypt(string temp,char playfair[5][5])
{
    string cipher="";
    int i,x1,y1,x2,y2;
    for(i=0;i<temp.length();i+=2)
    {
        x1=row(playfair,temp[i]);
        x2=row(playfair,temp[i+1]);
        y1=column(playfair,temp[i]);
        y2=column(playfair,temp[i+1]);
        if(x1==x2)
            cipher=cipher.append(1,playfair[x1][(y1+1)%5]).append(1,playfair[x2][(y2+1)%5]);
        else if(y1==y2)
            cipher=cipher.append(1,playfair[(x1+1)%5][y1]).append(1,playfair[(x2+1)%5][y2]);
        else
            cipher=cipher.append(1,playfair[x1][y2]).append(1,playfair[x2][y1]);
    }
    return cipher;
}
string decrypt(string cipher,char playfair[5][5])
{
    string temp="";
    int i,x1,y1,x2,y2;
    for(i=0;i<cipher.length();i+=2)
    {
        x1=row(playfair,cipher[i]);
        x2=row(playfair,cipher[i+1]);
        y1=column(playfair,cipher[i]);
        y2=column(playfair,cipher[i+1]);
        if(x1==x2)
            temp=temp.append(1,playfair[x1][(y1+4)%5]).append(1,playfair[x2][(y2+4)%5]);
        else if(y1==y2)
            temp=temp.append(1,playfair[(x1+4)%5][y1]).append(1,playfair[(x2+4)%5][y2]);
        else
            temp=temp.append(1,playfair[x1][y2]).append(1,playfair[x2][y1]);
```

```
    }
    return temp;
}
int main()
{
    char playfair[5][5];
    string key,ptext,temp,cipher;
    int ch;
    cout<<"Enter plain text : ";cin>>ptext;
    temp=constructDigrams(ptext);
    do
    {
        for(int i=0;i<5;i++)for(int j=0;j<5;j++)playfair[i][j]=0;
        cout<<"Enter the key    : ";cin>>key;
        constructPlayFairMat(playfair,key);
        for(int i=0;i<5;i++)
        {
            for(int j=0;j<5;j++)
                cout<<playfair[i][j]<<"\t";
            cout<<endl;
        }
        cipher=encrypt(temp,playfair);
        cout<<"Cipher text      : "<<cipher<<endl;
        temp=decrypt(cipher,playfair);
        cout<<"Decipher text    : "<<temp<<endl;
        cout<<"\n1 : Continue\n0 : Exit"<<endl;
        cin>>ch;
        cout<<endl;
    }
    while(ch==1);
    return 0;
}
```

Output :