```
/*
Copyright 2015 ALY SHMAHELL

 This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Input.H>
#include <FL/Fl_Output.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Text_Display.H>
#include <string>
#include <string.h>
#include <stdint.h>
#include "ENDABI_RSA_CORE.hpp"
#include "3rd_party/InfInt.h"
using namespace std;

/*Global Widgets*/

//Define Buffers
Fl_Text_Buffer *pubbuff = 0;
Fl_Text_Buffer *modbuff = 0;
Fl_Text_Buffer *prvbuff = 0;
Fl_Text_Buffer *encbuff = 0;
Fl_Text_Buffer *decbuff = 0;

//Define Display Boxes
Fl_Text_Display *private_key = 0;
Fl_Text_Display *public_key = 0;
Fl_Text_Display *Modulus = 0;
Fl_Text_Display *encrypted_message = 0;
Fl_Text_Display *decrypted_message = 0;

//Define Input Text Boxes
Fl_Input *desired_pubk = 0;
Fl_Input *d_p =0;
Fl_Input *d_q =0;
Fl_Input *mssg =0;

/**/

//global variables
InfInt desired_pk,desired_p,desired_q;
pub_key_sequence<InfInt> pks = {(InfInt)0,(InfInt)0,(InfInt)0,(InfInt)0};
int mssg_length;
InfInt encrypted_numerical[200];
InfInt decrypted_numerical[200];
InfInt prvk;

//Callback Function to generate both Public & Private Keys
void gen_keys(Fl_Widget*,void*)
{
    //Input
    desired_pk=(InfInt)(desired_pubk->value());
    desired_p =(InfInt)(d_p->value());
    desired_q =(InfInt)(d_q->value());
    //Public Key Generation
```

```
    pks = generate_public_key(desired_pk,desired_p,desired_q);
    //Private Key Calculation
    prvk = calculate_private_key(pks.pk,pks.p_,pks.q_);
    //Output Public Key
    {
        stringstream pks_strings;
        pks_strings<<pks.pk;
        string pks_string = pks_strings.str();
        pubbuff->text(pks_string.c_str());
        public_key->buffer(pubbuff);
    }
    //Output Modulus
    {
        stringstream pks_strings;
        pks_strings<<pks.n;
        string pks_string = pks_strings.str();
        modbuff->text(pks_string.c_str());
        Modulus->buffer(modbuff);
    }
    //Output Private Key
    {
        stringstream prvks_strings;
        prvks_strings<<prvk;
        string prvks_string = prvks_strings.str();
        prvbuff->text(prvks_string.c_str());
        private_key->buffer(prvbuff);
    }
}

/*Callback Function to Encrypt a message in the message Text Box
onto the Encrypted Message Display Box*/
void enc_mssg(Fl_Widget*,void*)
{
    //Input
    stringstream msgs;
    msgs<<mssg->value();
    string msg = msgs.str();
    const char* message = msg.c_str();
    //variables
    InfInt plain_numerical[200];
    mssg_length = strlen(message);
    stringstream encrypted_msgs;
    //Encryption
    for(int i =0; i<mssg_length; i++)
    {
        plain_numerical[i]=message[i]-'0';
        encrypted_numerical[i]=encrypt((InfInt)plain_numerical[i],pks.pk,pks.n);
        encrypted_msgs<<encrypted_numerical[i];
    }
    //Output
    string encrypted_msg = encrypted_msgs.str();
    encbuff->text(encrypted_msg.c_str());
    encrypted_message->buffer(encbuff);
}

/*Callback Function to Decrypt a message in the Encrypted Message Display Box
onto the Decrypted Message Display Box*/
void dec_mssg(Fl_Widget*,void*)
{
    stringstream decrypted_msgs;
    for(int i =0; i<mssg_length; i++)
    {
        decrypted_numerical[i]=decrypt(encrypted_numerical[i],prvk,pks.n);
        decrypted_msgs<<decrypted_numerical[i];
    }
    string decrypted_msg = decrypted_msgs.str();
    decbuff->text(decrypted_msg.c_str());
    decrypted_message->buffer(decbuff);
}

int main()
{
    //Initialize a Window
```

```
    Fl_Window *win =  new Fl_Window(990, 680, "ENDABI_RSA_DEMO_GUI");
    //Initialize Buffers
    pubbuff =  new Fl_Text_Buffer();
    modbuff =  new Fl_Text_Buffer();
    prvbuff =  new Fl_Text_Buffer();
    encbuff =  new Fl_Text_Buffer();
    decbuff =  new Fl_Text_Buffer();
    //Initialize Input Text Boxess
    desired_pubk= new Fl_Input(150,10,800,30,"Desired Public Key");
    d_p= new Fl_Input(150,50,800,30,"Desired p");
    d_q = new Fl_Input(150,90,800,30,"Desired q");
    mssg = new Fl_Input(450,370,200,30,"Message");
    //Initialize Display Boxes
    public_key = new Fl_Text_Display(150,200,800,30,"Public-Key");
    Modulus = new Fl_Text_Display(150,260,800,30,"Modulus");
    private_key = new Fl_Text_Display(150,310,800,30,"Private-Key");
    encrypted_message = new Fl_Text_Display(150,430,800,130,"Encrypted Message");
    decrypted_message = new Fl_Text_Display(450,640,200,30,"Decrypted Message");
    //Define & Initialize Buttons
    Fl_Button *generate_keys = new Fl_Button(480,140,150,30,"Generate Keys");
    Fl_Button *encrypt_message = new Fl_Button(150,370,150,30,"Encrypt Message");
    Fl_Button *decrypt_message = new Fl_Button(470,580,150,30,"Decrypt Message");
    //Link Buttons to callback functions
    generate_keys->callback(gen_keys);
    encrypt_message->callback(enc_mssg);
    decrypt_message->callback(dec_mssg);
    //Run Application
    win->show();
    //Return
    return(Fl::run());
}
```