TABLE OF CONTENTS

INTRODUCTION TO SOLUTION DESIGN

1.1 Introduction / Background

Boulevard Service Apartment's community parcel locker system is a menu-driven application program developed using C++ programming. All lockers are controlled by a centralized system. The centralized system allows residents of the Boulevard Service Apartment to register accounts, check profile information, collect parcels, and allows courier service to deposit parcels into lockers. Besides, an account is required for receiving one-time passcode to collect parcels from lockers. Thus, apartment unit number, full name, contact number, and password are required for account registra,tion and users' account passwords will be encrypted to protect users' data.

Assumptions

1. Boulevard Service Apartment has 9 floors and 10 units each floor.
2. The first digit of the apartment unit number represents the floor number (Floor 1-9), while the second digit represents the house number (Unit 0-9).
3. Apartment unit number ranges from 10-99.
4. The community parcel locker consists of 3 blocks of lockers with 10 lockers on each block
5. The first block of the community parcel locker's locker ID ranges from 1 - 10, the second block ranges from 11-20, and the third block ranges from 21-30.
6. One apartment unit can register multiple accounts under different names and phone numbers.
7. Only one parcel will be allowed to be deposited under one apartment unit.
8. Deletion and modification for existing users' information are not needed.
9. Customer service representatives will help users reset passwords.
10. Customer service representatives will provide assistance to users of Boulevard Service Apartment's community parcel locker system.

1.2 System Roles / Users

Boulevard Service Apartment's community parcel locker system end users are Boulevard Service Apartment residents and courier service (delivery men).

Residents will be able to:

1. Register an account.
2. Login account.
3. Check profile information (apartment unit number, username and contact number).
4. Check parcel locker number and one-time passcode.
5. Collect parcels.

Courier service will be able to:

1. Deposit parcel by entering receiver's unit number into the system.

1.3 System Modules

1.3.1 Module 1: Resident Registration

1. This module is for new users (Boulevard Service Apartment residents) to register accounts before they use other functions in this parcel locker system.
2. This module requires user inputs such as their name, contact number, unit number, and password.
3. Contact number will be validated to ensure it contains only digits.
4. Before getting the unit number, the system will compare the name and contact number to those in the database. If either one of them is found duplicated, it'll be considered as an existing user and, therefore, not allowed to register again.
5. Then, the range of unit numbers will be validated.
6. Before the data is saved into the database, the user's password will be encrypted into a garbled output to protect the user's information.
7. After that, users will be directed to the main menu.

1.3.2 Module 2: User Login

1. This module allows existing users to log in to their information page and check the locker number and the OTP(if any).
2. This module requires only two user inputs which are their name and password.
3. While the user is typing the password, asterisks will show on the screen instead of the actual password.

4. If both the username and password match with that in our database, the user will be directed to the user information page.
5. Meanwhile, if the condition above does not meet or the user has not registered yet, an error message will be displayed, and the user may choose to re-login or back to the main menu.

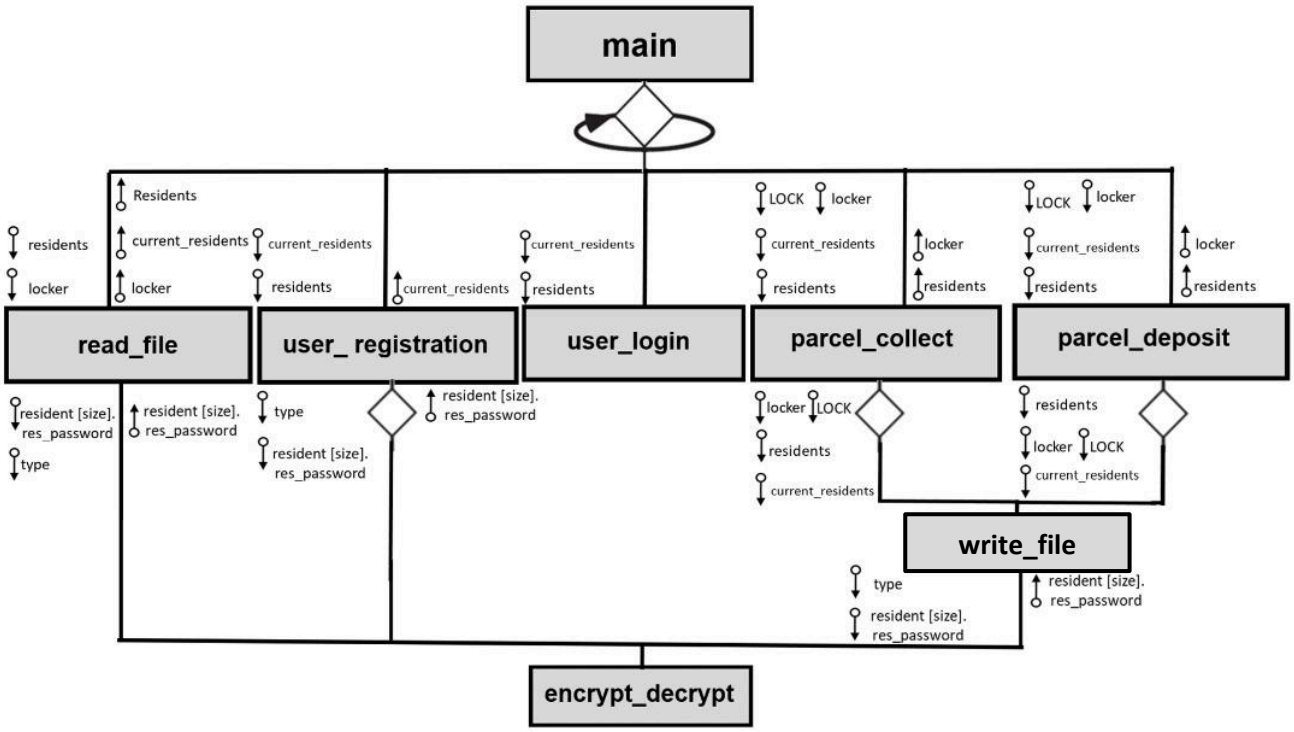
1.3.3 Module 3: Parcel Collection

1. This module allows users (residents) to collect their parcels.
2. Therefore, the user inputs locker numbers, and OTP is required.
3. The locker number will be validated to ensure it is within the range and occupied with a parcel.
4. If the locker is empty, the user may choose to exit this function as the parcel might be collected by their family or housemate who shares the same unit number, so the user may choose to exit this function. This is to prevent them from sticking in the loop.
5. Then, OTP will be read and compared with the locker's OTP.
6. If it matches, the locker's OTP, resident's locker number, and OTP will all be erased.
7. If not, the users may re-enter the OTP or exit the function.

1.3.4 Module 4: Parcel Deposition

1. This module is for delivery men to store parcels.
2. Inputs of unit number and locker number is required.
3. The unit number will be validated to ensure that:
    a. The unit number is within the range.
    b. The resident of the unit number had registered.
    c. The unit number has not registered any parcel yet. Though we assume 1 unit will only deposit 1 parcel, it might happen that the delivery men key in wrongly. So, if that happens, the delivery man will have to key in again or exit the function.
4. To ensure the range of locker number and the locker is not occupied with any parcel, it will be validated.
5. If all the conditions meet, OTP will be generated and also validated to ensure no duplication.
6. Then the data will be stored into both locker's and residents' databases.

1.4 Structure Chart



1.5 Pseudocode

Main Module

START

Loop the menu if the user does not select exiting function by using do-while loop

Retrieve and input the information of residents.

The cover page of menu displayed

Ask user to select the next action

By using switch loop

When the user enters 1, the program will retrieve the registration interface.

When user enters 2, the program will retrieve the login interface for the user to get a one-time PIN for the parcel collection.

When the user enters 3, the program will retrieve the parcel interface.

When the user enters 4, the program will retrieve the parcel deposition, and this is only used for courier.

When the user enters 5 is for exiting the system.

Others entered instead of these 5 options will be displaying an ERROR message.

Before exiting the system, the CLOSE message will be displayed.

END

Module i: User Registration

START

User registration interface displayed.

Read username and contact number.

Validate whether the contact number is digit or the user had registered before.

Exit this module and display an error message when the name and contact number is matched with the database.

Else, unit number and password will be read.

Validate unit number to make sure it is within range 10-99.

Password will be encrypted and display '*'.

All the data will be written into the resident_data file.

Display "Register Successfully".

STOP

Module ii: User Login

START

Loop this module when the user wants to login again.

Display the login interface.

Read username and password.

Passwords will not be shown while the user key in. Instead, '*' will be displayed.

If the name and password entered is matched with each other. User information page will be displayed.

Else, ERROR message will be displayed.

Read user' option whether he or she wants to re-login.

STOP

Module iii: Parcel Collection

START

Display the header of parcel collection.

Read locker number and validate the range and status of locker.

If locker number is outside the range, users will be required to re-enter.

If the locker is empty, users may choose to exit the function or loop back to re-enter locker number.

If all the conditions above are met, the system will continue to read OTP.

If OTP does not match, an error message will be displayed and the user may choose to re-enter OTP or exit the function.

Else, a valid message will be displayed.

The locker OTP array, residents' locker number and OTP array will be erased.

All the new data will be written into both resident and locker text files.

STOP

Module iv : Parcel Deposition

START

Display the header of the parcel deposition interface.

Read unit number.

Validate to ensure the range of unit number and whether the resident had registered in this system.

While it's not within the range 10-99, loop back to reread the unit number.

If the user has not registered yet, an error message will be displayed and the user may choose to re-enter or exit the function.

Else, if the unit number has registered any parcel, An error message will be displayed, and the user may choose to re-enter or exit the function.

Else, read the locker number.

First validate the range followed by the status of the locker.

If it is outside the range or the locker is full, read the locker number again.

Generate OTP.

Validate if there is any duplicate OTP in our database.

If yes, loop back to generate OTP again.

Else, OTP will be stored into the locker and resident arrays. The locker number will be stored into the resident arrays as well.

Write into both locker and resident text files.

Loop this module when the courier wants to continue to deposit any parcel.

STOP

C++ PROGRAM

```
#include <iostream>
#include <cctype>
#include <fstream>
#include <conio.h>
#include <string.h>
#include <cstdlib>
#include <ctime>
using namespace std;

#define LOCK 30          //Maximum locker.
#define MAX_SELECTION 5 //We have currently 5 main functions only.
#define MAX_RESIDENT 200 //Maximum resident records
#define KEY 0xFACA //secret key for cryptography.

typedef struct
{
        char res_name[50];
        int  res_unit_no;
        char res_password[15];
        char res_contact[12];
        int  lock_id;
        char res_otp[6];

} USER_DATA;   //Details of the registered resident.

typedef struct
{
        int        id;
        char lock_otp[6];
        bool lock_status;

} LOCKER;      //Lockers status and OTP.

int read_file(USER_DATA* residents, LOCKER* locker);   //Read resident and locker data.
int user_registeration(USER_DATA* residents, int size);   //For new user to register.
void user_login(USER_DATA* residents, int size);    //For existing user to check otp and own account
details.
void parcel_collect(USER_DATA* residents, int size, LOCKER* locker, int lock);   //To match otp and
collect parcel.
void parcel_deposit(USER_DATA* residents, int size, LOCKER* locker, int lock);   //For delivery man to
load parcel.
void encrypt_decrypt(char* password, char types);
void write_file(USER_DATA* residents, int size, LOCKER* locker, int lock);

int main()
{
        char function;
```

```
	do {
		USER_DATA residents[MAX_RESIDENT] = { 0 };
		LOCKER locker[LOCK] = { 0 };
		int current_resident = 0;
		current_resident = read_file(residents, locker);

		cout << "\n
************************************************************\n";
		cout << " *                                          *\n";
		cout << " *                  WELCOME!                *\n";
		cout << " *                                          *\n";
		cout << " *  1.New User Registration                  *\n";
		cout << " *  2.Login                                 *\n";
		cout << " *  3.Collect Parcel                        *\n";
		cout << " *  4.Deposit Parcel (Courier use only)     *\n";
		cout << " *  5.Exit                                  *\n";
		cout << " *                                          *\n";
		cout << " *                                          *\n";
		cout << "
************************************************************\n";

		cout << "\n Please enter the number for your choice: ";
		cin >> function;

		switch (function) {
		case '1': {		system("CLS");		current_resident =
user_registeration(residents, current_resident);	break; }
		case '2': {		system("CLS");		user_login(residents, current_resident);
	break; }
		case '3': {		system("CLS");		parcel_collect(residents, current_resident,
locker, LOCK);	break; }
		case '4': {		system("CLS");		parcel_deposit(residents,
current_resident, locker, LOCK);	break; }
		case '5': {		break;
			 }
		default: {
			cout << "\n ERROR: Input only valid between " << 1 << " and " <<
MAX_SELECTION << endl << endl;
			system("pause");
			break;
		}
		}
		system("CLS");

	} while (function != '5');

	cout << "\n ************************************************************\n";
	cout << " *                                          *\n";
```

```
	cout << " *                                          *\n";
	cout << " *                                          *\n";
	cout << " *          THANK YOU FOR USING OUR SERVICE!            *\n";
	cout << " *             HOPE TO SEE YOU AGAIN!                *\n";
	cout << " *                                                  *\n";
	cout << " *                                          *\n";
	cout << " *                                          *\n";
	cout << " ************************************************************\n";

	return 0;
}

int read_file(USER_DATA* residents, LOCKER* locker)
{
	int size = 0, lock = 0;
	ifstream in_file("resident_data.txt");

	if (!in_file)
		cout << "Error input file open (resident_data.txt)\n";
	else
	{
		in_file.getline(residents[size].res_name, 50);    //Get name from resident_data.txt
		while (in_file)
		{
			in_file.getline(residents[size].res_password, 15);
			encrypt_decrypt(residents[size].res_password, 'd');    //Decrypt the password
after getting it from the file.

			in_file.getline(residents[size].res_contact, 12);
			in_file >> residents[size].res_unit_no;
			in_file >> residents[size].lock_id;
			in_file >> residents[size].res_otp;
			size++;

			if (in_file.peek() == '\n')
				in_file.ignore(256, '\n');

			in_file.getline(residents[size].res_name, 50);
		}
		in_file.close();
	}

	ifstream in_file2("locker.txt");

	if (!in_file2)
		cout << "Error input file open (locker.txt)\n";
	else
	{
```

```
		in_file2 >> locker[lock].id;
		while (in_file2) //Get data from locker.txt
		{
			string temp;
			in_file2 >> temp;
			if (temp == "true")
				locker[lock].lock_status = true;
			else
				locker[lock].lock_status = false;

			in_file2 >> locker[lock].lock_otp;
			lock++;

			if (in_file2.peek() == '\n')
				in_file2.ignore(256, '\n');

			in_file2 >> locker[lock].id;
		}
		in_file2.close();
	}

	return size;
}

int user_registeration(USER_DATA* residents, int size)
{
	int name_len, pass_len, cont_len, unit_no;
	char name[50], contact[12];
	int i = 0;
	char word = ' ';

	cout << "\n ********************** USER REGISTRATION **********************\n";
	while (getchar() != '\n');
	cout << "\n     Name: ";
	cin.getline(name, 50);
	name_len = strlen(name);   //Get the length of name.

	bool digit = true;
	do {
		cout << "\n     Contact number (without '-'): ";
		cin.getline(contact, 12);
		cont_len = strlen(contact);    //Get the length of contact.

		for (int z = 0; z < cont_len; z++) {    //Check if contact number input is valid.
			if (!isdigit(contact[z])) {
				digit = false;
				cout << "\n ERROR: Input only can only consist numerics (0 to 9)" << endl
<< endl;
```

```
                        break;
                }
                else digit = true;
        }
    } while (!digit);

    bool same_user = false;     //Check if the user register before.
    for (int j = 0; j < size; j++) {
        if (strcmp(name, residents[j].res_name) == 0 || strcmp(contact, residents[j].res_contact)
== 0)
        {
            cout << "\n This user had registered before.\n Please call our customer service
for any enquiries.\n";
            same_user = true; break;
        }
        else
            same_user = false;
    }

    if (!same_user) {
        size++;
        strcpy_s(residents[size].res_name, name);
        for (int c = 0; c < 12; c++)
            residents[size].res_contact[c] = contact[c];
        do {
            cout << "\n     Unit number : ";
            cin >> residents[size].res_unit_no;

            if (residents[size].res_unit_no < 10 || residents[size].res_unit_no > 99)     //Unit
number validation.
                cout << "\n ERROR: Invalid unit number (Unit no. between 10 to
99)\n\n";

        } while (residents[size].res_unit_no < 10 || residents[size].res_unit_no > 99);

        cout << "\n     Password : ";
        while (word != 13) {
            word = _getch();
            if (word != 13) {
                residents[size].res_password[i] = word;
                cout << "*";
                i++;
            }
        }
        pass_len = strlen(residents[size].res_password);

        residents[size].lock_id = 0;
        strcpy_s(residents[size].res_otp, "0");
```

```
		ofstream out_res;
		out_res.open("resident_data.txt", ios::app);
		if (!out_res)
			cout << " Errors opening resident files";
		else {	//Write new resident information into resident_data.txt
			out_res << "\n";
			out_res.write(residents[size].res_name, name_len) << "\n";
			encrypt_decrypt(residents[size].res_password, 'e');
			out_res.write(residents[size].res_password, pass_len) << "\n";
			out_res.write(residents[size].res_contact, cont_len) << "\n";
			out_res << residents[size].res_unit_no << "\n"
				<< residents[size].lock_id << "\n"
				<< residents[size].res_otp;

			encrypt_decrypt(residents[size].res_password, 'd');

			cout << "\n\n Registered successfully!\n";
		}
		out_res.close();
	}
	system("Pause");

	return size;
}

void user_login(USER_DATA* residents, int size)
{
	int  i;
	char name[50] = " ", password[15] = "", con_log = 'n';
	do {
		cout << "\n ************************* USER LOGIN
*************************\n";

		bool match = false;
		while (getchar() != '\n');
		cout << "\n	Enter User Name: ";
		cin.getline(name, 50);

		cout << "\n	Enter Password : ";
		int z = 0;
		char word = ' ';
		while (word != 13) {
			word = _getch();

			if (word != 13) {
				password[z] = word;
				cout << "*";
```

```
                        z++;
                }
        }

        for (i = 0; i < size; i++)
        {
                if (strcmp(password, residents[i].res_password) == 0 && strcmp(name,
residents[i].res_name) == 0)    //Username and password validation.
                {
                        system("CLS");
                        cout << "\n ********************** Successfully log in
**********************\n\n";
                        cout << "     Name :" << residents[i].res_name << endl;
                        cout << "\n    Contact no :" << residents[i].res_contact << endl;
                        cout << "\n    Resident Unit :" << residents[i].res_unit_no << endl;
                        cout << "\n    Locker number :" << residents[i].lock_id << endl;
                        cout << "\n    Otp :" << residents[i].res_otp << endl << endl;
                        match = true;
                        break;
                }
        }
        if (!match)
                cout << endl << "\n Incorrect unit number or password./ User doesn't exist.";

        cout << "\n Enter Y to login again(Any key to exit) : ";
        cin >> con_log;
        system("CLS");

    } while (con_log == 'y' || con_log == 'Y');

    return;
}

void parcel_collect(USER_DATA* residents, int size, LOCKER* locker, int lock)
{
    int lock_num, l;
    char otp[7], con_col = 'n';
    bool match = false, valid = true;
    cout << "\n ********************** PARCEL COLLECTION **********************\n";
    do {
        con_col = 'n';
        do {
            cout << "\n    Locker number (1-30):";
            cin >> lock_num;
            if (lock_num < 1 || lock_num > 30)      //Check if locker number input is correct.
                cout << "\n    Invalid locker number.Please re-enter.";
        } while (lock_num < 1 || lock_num > 30);
```

```
		if (locker[lock_num - 1].lock_status) {
			cout << "\n Locker are empty\n Enter Y to re-enter. (any key to exit): ";
//Check locker status.
			cin >> con_col;
			valid = false;
		}
		else
			valid = true;
	} while (con_col == 'y' || con_col == 'Y');

	if (valid)
	{
		lock_num -= 1;	//locker index number.
		do
		{
			con_col = 'n';
			cout << "\n    OTP:";
			cin >> otp;
			if (strcmp(locker[lock_num].lock_otp, otp) != 0)
			{
				cout << "\n OTP do not match.\n Enter Y to re-enter. (any key to exit):";
				cin >> con_col;
			}
			else
			{
				match = true;
				for (int i = 0; i < size; i++)
				{
					if (strcmp(residents[i].res_otp, otp) == 0)
					{
						residents[i].lock_id = 0;		//reset resident's otp
						strcpy_s(residents[i].res_otp, "0");
					}
				}
				locker[lock_num].lock_status = true;	//reset locker's status and otp
				strcpy_s(locker[lock_num].lock_otp, "0");

				write_file(residents, size, locker, LOCK);

				cout << "\n OTP matched.Please take your parcel.\n Thank you.\n";
				system("pause");
			}
		} while (con_col == 'y' || con_col == 'Y');
	}
}

void parcel_deposit(USER_DATA* residents, int size, LOCKER* locker, int lock)
{
```

```
	int unit_num, lock_num;
	char con_dep = 'n';

	cout << "\n ********** PARCEL DEPOSITION **********\n";
	do {
		cout << "\n   Please enter unit number : ";
		do {
			cin >> unit_num;
			if (unit_num < 10 || unit_num > 99)	//check if unit number input is within range.
				cout << "\n Invalid unit number.Please re-enter:";

		} while (unit_num < 10 || unit_num > 99);

		bool found = false;
		int unit_index = 0;
		for (int i = 0; i < size; i++)	//check if the resident had register.
		{
			if (unit_num == residents[i].res_unit_no)
			{
				found = true;
				unit_index = i;
			}
		}

		if (!found)
			cout << "\n ERROR: Owner of unit " << unit_num << " have not register an account.\n Enter Y to re-enter (any key to exit): ";
		else
		{
			if (residents[unit_index].lock_id != 0)
				cout << "\n One unit only allowed to deposit one parcel.\n Enter Y to re-enter (any key to exit): ";
			else {
				lock_num = 1;
				do {
					cout << "\n   Please enter locker number (1-30):";
					cin >> lock_num;

					if (lock_num < 1 || lock_num > 30)	//check if locker number input is within range.
						cout << "\n Invalid locker number.Please re-enter:";
					else if (!locker[lock_num - 1].lock_status)	//check locker status.
						cout << "\n Locker are full.\n Please choose another locker:";
```

```
                                } while (!locker[lock_num - 1].lock_status || (lock_num < 1 ||
lock_num > 30));

                                int l = 0;
                                char otp[6];
                                bool repeat = false;
                                do
                                {
                                        srand(time(NULL));

                                        for (int i = 0; i < 6; i++)
                                                otp[i] = rand() % 10 + '0';

                                        for (l = 0; l < LOCK; l++)   //Check if there is any repeated otp.
                                        {
                                                if (strcmp(otp, locker[l].lock_otp) == 0) {
                                                        repeat = true;
                                                        break;
                                                }
                                        }

                                } while (repeat);

                                for (int s = 0; s < size; s++)   //Check account that stored the same unit
number then store otp and locker number.
                                {
                                        if (unit_num == residents[s].res_unit_no)
                                        {
                                                residents[s].lock_id = lock_num;
                                                for (int z = 0; z < 6; z++)
                                                        residents[s].res_otp[z] = otp[z];
                                        }
                                }

                                locker[lock_num - 1].lock_status = false;    //Store otp to corresponding
locker.
                                for (int z = 0; z < 6; z++)
                                        locker[lock_num - 1].lock_otp[z] = otp[z];

                                write_file(residents, size, locker, LOCK);

                                cout << "\n Parcel loaded successfully.\n Continue parcel deposition? (Y
for YES/ any key to exit): ";
                        }
                }
                cin >> con_dep;
        } while (con_dep == 'y' || con_dep == 'Y');
}
```

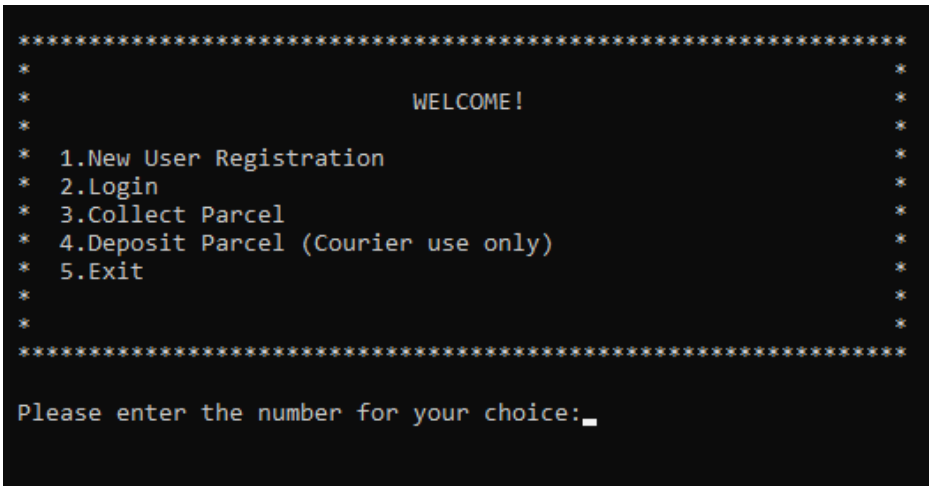
```
void encrypt_decrypt(char* password, char types) {

	int i;
	//encryption
	if (types == 'e') {
		for (i = 0; i < strlen(password); ++i)
		{
			password[i] = password[i] - KEY;
		}
	}
	//decryption
	else if (types == 'd') {
		for (i = 0; i < strlen(password); ++i)
		{
			password[i] = password[i] + KEY;
		}
	}
	else
		cout << "\n Error : invalid operation (encryption/decryption)" << endl;
}

void write_file(USER_DATA* residents, int size, LOCKER* locker, int lock)
{
	ofstream out_lock;
	ofstream out_res;
	out_lock.open("locker.txt", ios::out);
	out_res.open("resident_data.txt", ios::out);

	if (!out_lock)
		cout << " Errors opening locker files";
	if (!out_res)
		cout << " Errors opening resident files";

	int name_len, pass_len, cont_len;
	for (int i = 0; i < size; i++)
	{
		name_len = strlen(residents[i].res_name);
		pass_len = strlen(residents[i].res_password);
		cont_len = strlen(residents[i].res_contact);

		if (i != 0)
			out_res << "\n";
		out_res.write(residents[i].res_name, name_len) << "\n";
		encrypt_decrypt(residents[i].res_password, 'e');    //Encrypt the password before
writing it to the file.
		out_res.write(residents[i].res_password, pass_len) << "\n";
		out_res.write(residents[i].res_contact, cont_len) << "\n";
```

```
            out_res << residents[i].res_unit_no << "\n"
                    << residents[i].lock_id << "\n"
                    << residents[i].res_otp;

            encrypt_decrypt(residents[i].res_password, 'd');    //Decrypt the password.
        }
        out_res.close();

        for (int j = 0; j < LOCK; j++)
        {
            if (j != 0)
                out_lock << "\n";

            string status;
            out_lock << locker[j].id << "\n";
            if (locker[j].lock_status == true)
                status = "true";
            else
                status = "false";
            out_lock << status << "\n";
            out_lock << locker[j].lock_otp;
        }
        out_lock.close();
}
```

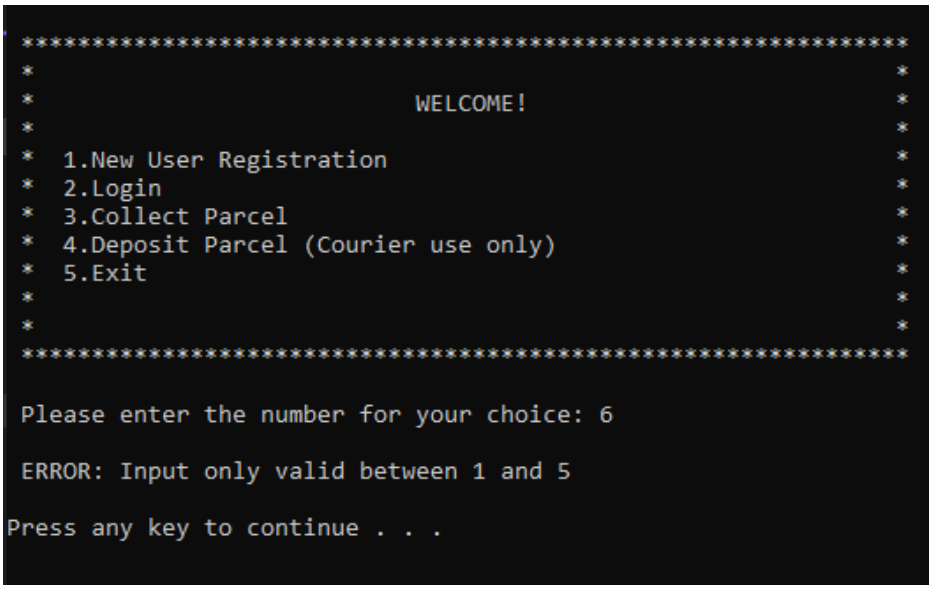
SAMPLE OUTPUT

3.0 *Main menu*

1. This is the first output on the screen when the user runs the program.



2. When user input is other than 1/2/3/4/5 as has been stated, the input will be invalid. The screen will show that it is an invalid input and ask the user to re-enter.



3.1 *Module 1:Resident Registration*

New User Registration (1) selection in the main menu will bring the user to here and ask them to enter their name followed by other details.

3.1.1 Register successfully- Save to file and encrypted password.

1. From here onwards, we use 'Lee Chin Cheong' as an example. If there is no error, the user data will be saved successfully. The password here is '0987654321', but it will show '*' and encrypted before saving into the resident data.

```
********** USER REGISTERATION **********

    Name: Lee Chin Cheong

    Contact number (without '-'): 0189987543

    Unit number : 66

    Password : **********

 Registered successfully!
Press any key to continue . . .
```

```
137224
Lee Chin Cheong
fonmlkjihg
0189987543
66
0
0
```

3.1.2 Contact number and Unit number validation

1. If the user key in the invalid contact number or unit number, the program will be shown as below.

```
********** USER REGISTERATION **********

    Name: Lee Chin Cheong

    Contact number (without '-'): 0189987t54

ERROR: Input only can only consist numerics (0 to 9)


    Contact number (without '-'): 0189987543

    Unit number : 166

ERROR: Invalid unit number (Unit no. between 10 to 99)


    Unit number : 66

    Password : **********

 Registered successfully!
Press any key to continue . . .
```

3.1.3 Existing user re-registration (invalid)

1. If the user with the same name or contact number has had an account, the user will not be able to register anymore. Then, the system will direct the user back to the main menu.

```
********** USER REGISTERATION **********

    Name: Lee Chin Cheong

    Contact number (without '-'): 0189987543

This user had registered before.
Please call our customer service for any enquiries.
Press any key to continue . . .
```

3.2 *Module 2:Resident Login*

User Login(2) selection in the main menu will bring the user to here and ask them to enter their name and password to login to their account.

3.2.1 Match account username and decrypted password.

1. All the users' passwords will be decrypted before the system reads the resident data file. When the users key in the wrong password, they may choose to exit or try again.

```
********** USER LOGIN **********

    Enter User Name: Lee Chin Cheong

    Enter Password : ***********

Incorrect unit number or password./ User doesn't exist.
Enter Y to login again(Any key to exit) : y

    Enter User Name: Lee Chin Cheong

    Enter Password : **********
```

3.2.2 User Information Page

1. The User Information Page will be shown if the password matched.

```
********** Successfully log in **********

    Name :Lee Chin Cheong

    Contact no :0189987543

    Resident Unit :66

    Locker number :0

    Otp :0

Press any key to continue . . .
```

3.3 *Module 3: Parcel Collection*

Parcel Collection (3) selection in the main menu will bring the user to here and ask them to enter locker number and OTP.

3.3.1 Invalid locker number - not between 1 to 30.

1. Locker number and locker status will be validated before users are allowed to key in the otp.

```
********** PARCEL COLLECTION **********

    Locker number (1-30):33

    Invalid locker number.Please re-enter.
```

3.3.2 Empty locker

1. Users will have to re-enter the locker number if it is not between 1 to 30. The system will direct the user to exit this function, if the locker is empty.

```
    Locker number (1-30):13

Locker are empty
Enter Y to re-enter. (any key to exit): _
```

3.3.3 Parcel collection - valid & invalid OTP.

1. If the otp does not match, users may choose to re-enter the otp or exit the system.

```
********** PARCEL COLLECTION **********

    Locker number (1-30):2

    OTP:425261

OTP do not match.
Enter Y to re-enter. (any key to exit):y

    OTP:428809

OTP matched.Please take your parcel.
Thank you.
Press any key to continue . . . _
```

3.4 *Module 4: Parcel Registration*

3.4.1 Unit number validation.

1. If the unit had deposited a parcel, it would not be allowed to deposit another parcel.
2. If the unit number is not between 10 to 99 or the resident hasn't registered an account in the system. Error message will be displayed and prompt the user to re-enter / exit the function.

```
********** PARCEL DEPOSITION **********

  Please enter unit number : 32

One unit only allowed to deposit one parcel.
Enter Y to re-enter (any key to exit):
```

```
********** PARCEL DEPOSITION **********

  Please enter unit number : 188

Invalid unit number.Please re-enter:83

ERROR: Owner of unit 83 have not register an account.
Enter Y to re-enter (any key to exit):
```

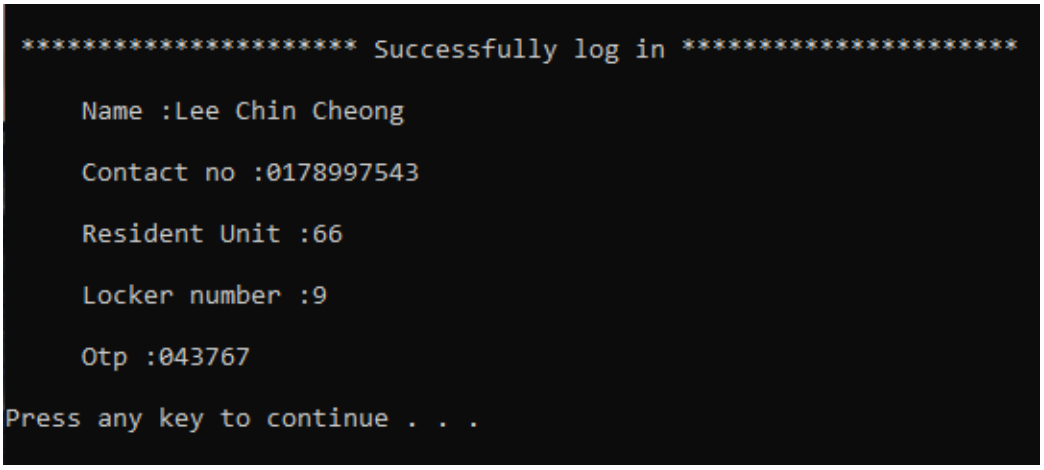
3.4.2 Locker number validation.

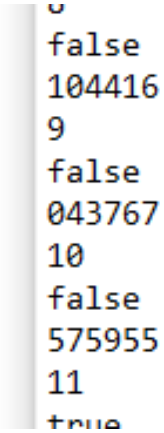
1. If the locker number is not between 1 to 30 or the locker is full. The delivery man will have to choose another locker number or exit.

```
********** PARCEL DEPOSITION **********

  Please enter unit number : 66

  Please enter locker number (1-30):33

Invalid locker number.Please re-enter:
  Please enter locker number (1-30):1

Locker are full.
Please choose another locker:
  Please enter locker number (1-30):9

Parcel loaded successfully.
Continue parcel deposition? (Y for YES/ N for NO): _
```

3.4.3 Parcel loaded successfully - Check otp at user information page.

1. When the parcel is loaded successfully, Otp will be stored in both resident data and locker data.
2. Then, users may login to their account to check the locker number and the corresponding otp.





3.5 *Exit Function*

If the Exit (5) in the main menu is selected, this output will be shown and then the user will exit the system.

SAMPLE INPUT

4.1 Text file 1 (Resident Data)

Name	Phone Number	Unit Number	Decrypted Password	Encrypted Password	Locker	OTP
Elton Lee	0179911233	10	abc123456	—˜™ghijkl	2	170435
Eve Lee	0176543678	99	bcd234567	˜™šhijklm	4	137224
Genevieve	0135670123	60	cde345678	™š›ijklmn	3	145497
Dato Sri Najib Razak	0187654367	10	def456789	š›œjklmno	0	0
Lee Yong Yee	0107654376	59	efg567890	›œklmnof	1	307880
Lim Han Yen	0128765437	44	fgh678901	œžlmnofg	0	0
Lee Chin Yee	0139876543	12	ghi789012	žŸmnofgh	8	104416
Lee Khang Ming	0149876567	19	hij890123	žŸ nofghi	5	062869
Lee Jia Wei	0159876543	77	ijk901234	Ÿ ¡ofghij	11	640407
Bryson Lew	0169876545	88	jkl012345	¡¢fghijk	10	575955
Lee Chong Wei	0170987654	23	klm123456	¡¢£ghijkl	0	0
Lee Zhi Jia	0180987654	73	lmn234567	¢£¤hijklm	0	0
Xi Jin Ping	0199876543	86	mno345678	£¤¥ijklmn	0	0
Chua Eng Wen	0109876543	37	nop456789	¤¥¦jklmno	0	0
Nancy Pelosi	0119876539	91	opq567890	¥¦§klmnof	0	0
Donald Trump	0128765450	51	pqr678901	¦§¨lmnofg	0	0
Joe Biden	0138765543	46	qrs789012	§¨©mnofgh	0	0
Boris Johnson	0145279219	68	rst890123	¨©ªnofghi	0	0
Chean Swee Ling	0153673891	28	stu901234	©ª«ofghij	0	0
Sor Kean Vee	0165645428	98	tuv012345	ª«¬fghijk	0	0
Khor Kok Chin	0173673690	35	uvw123456	«¬-ghijkl	0	0
Maryam Khanian Najafabadi	0186683519	66	vwx234567	¬-®hijklm	9	043767
Lee Thiam Ken	0122095551	99	5551	fgh>kkkg	4	137224

4.2 Text file 2 (Locker)

Locker ID	Status	OTP
1	false	307880
2	false	170435
3	false	145497
4	false	137224
5	false	062869
6	true	0
7	true	0
8	false	104416
9	false	043767
10	false	575955
11	false	640407
12	true	0
13	true	0
14	true	0
15	true	0
16	true	0
17	true	0
18	true	0
19	true	0
20	true	0
21	true	0
22	true	0
23	true	0
24	true	0
25	true	0

26	true	0
27	true	0
28	true	0
29	true	0
30	true	0