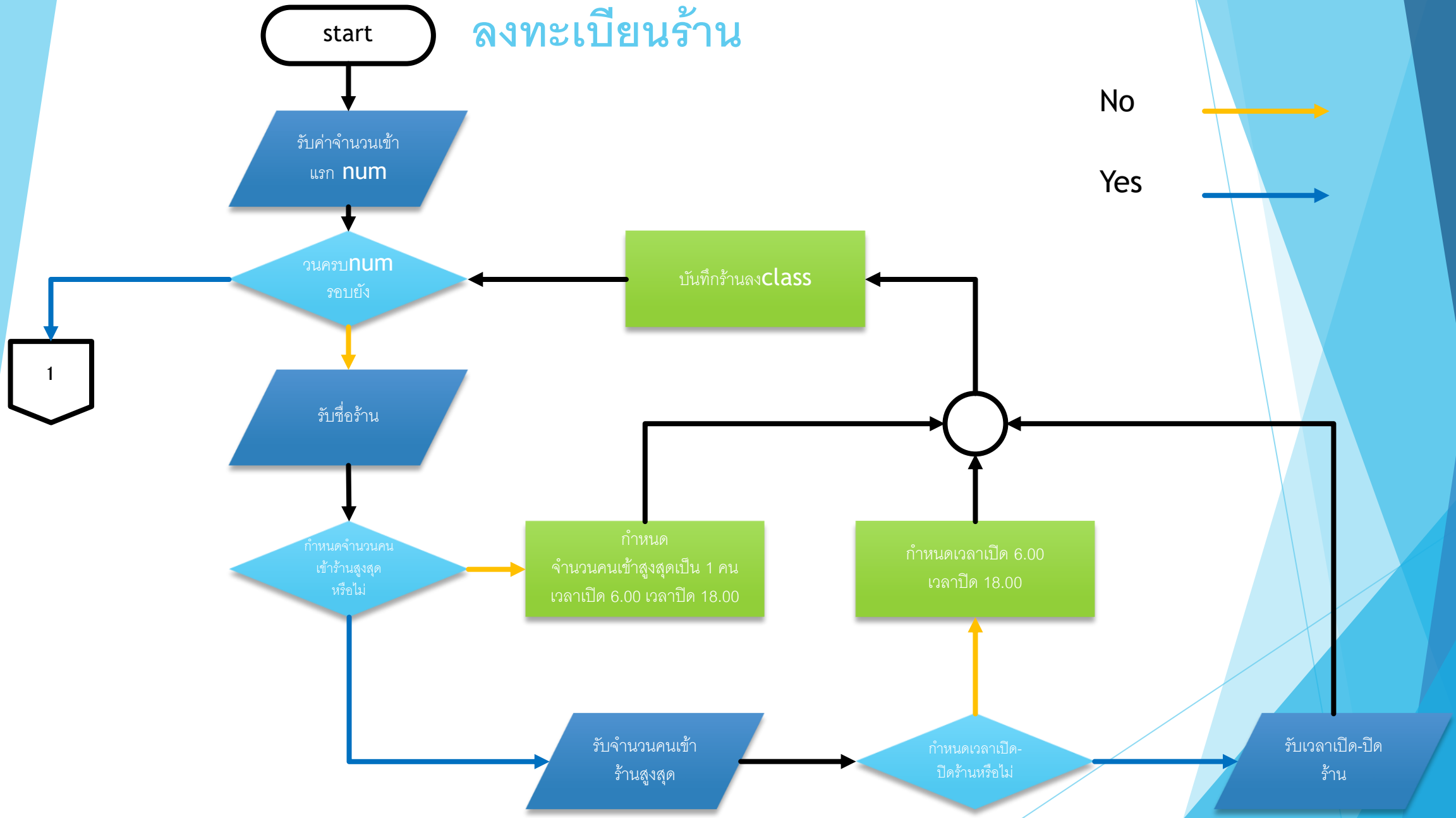
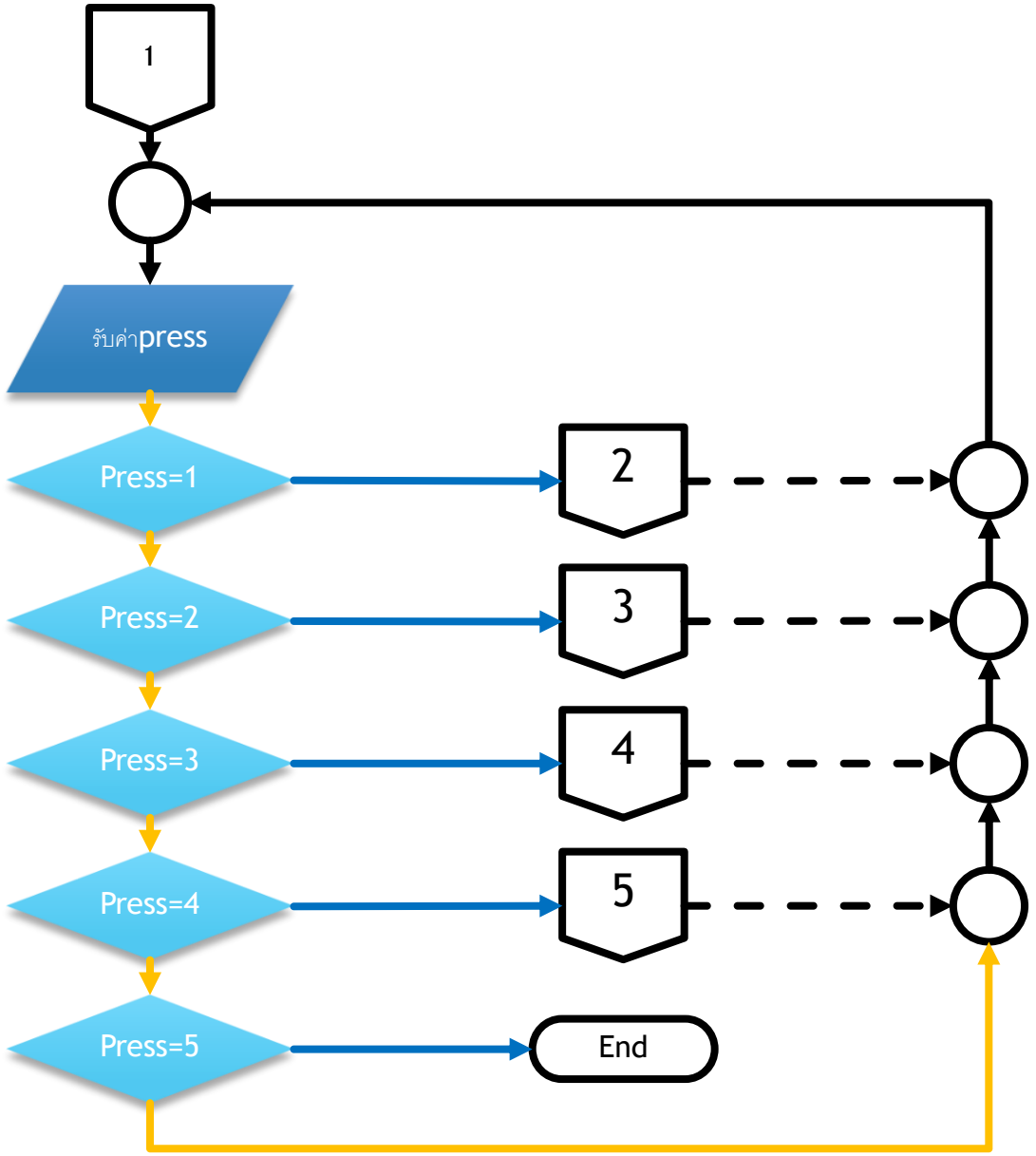
ระบบเวลาในการออกจากบ้านของแต่ละครอบครัวในช่วงปิดเมือง

แรงจูงใจ

- จากการแพร่ระบาดของcovid-19ทำให้ผู้คนต้องอยู่แต่ในบ้าน
- รัฐบาลไม่ต้องการให้มีผู้คนอยู่ในที่สาธารณะเป็นจำนวนมาก
- แต่ละร้านมีการจำกัดคนที่จะอยู่ในร้าน
- ต้องการติดตามการเคลื่อนไหวของแต่ละครอบครัวในที่สาธารณะ

ลงทะเบียนร้าน



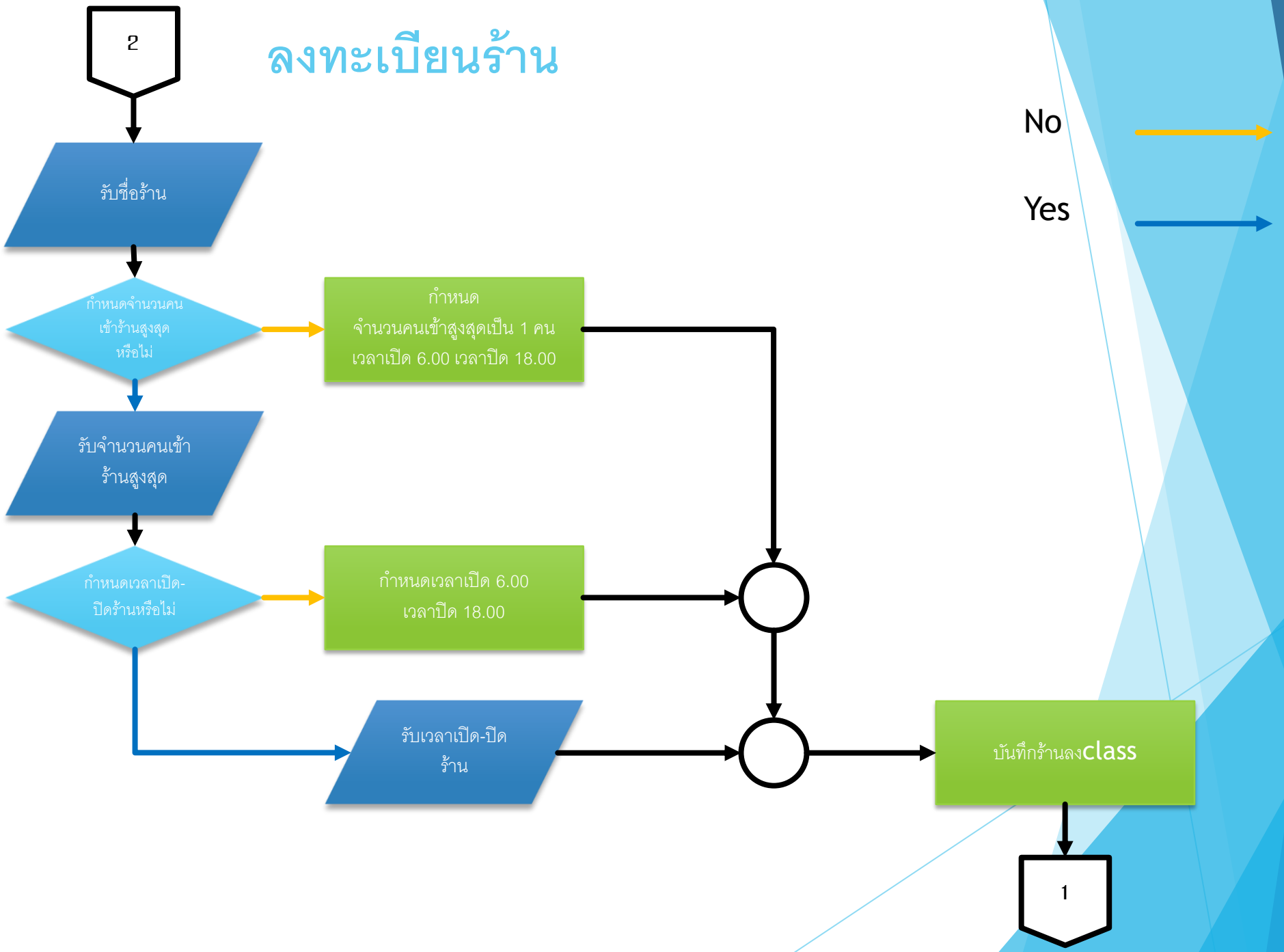


No

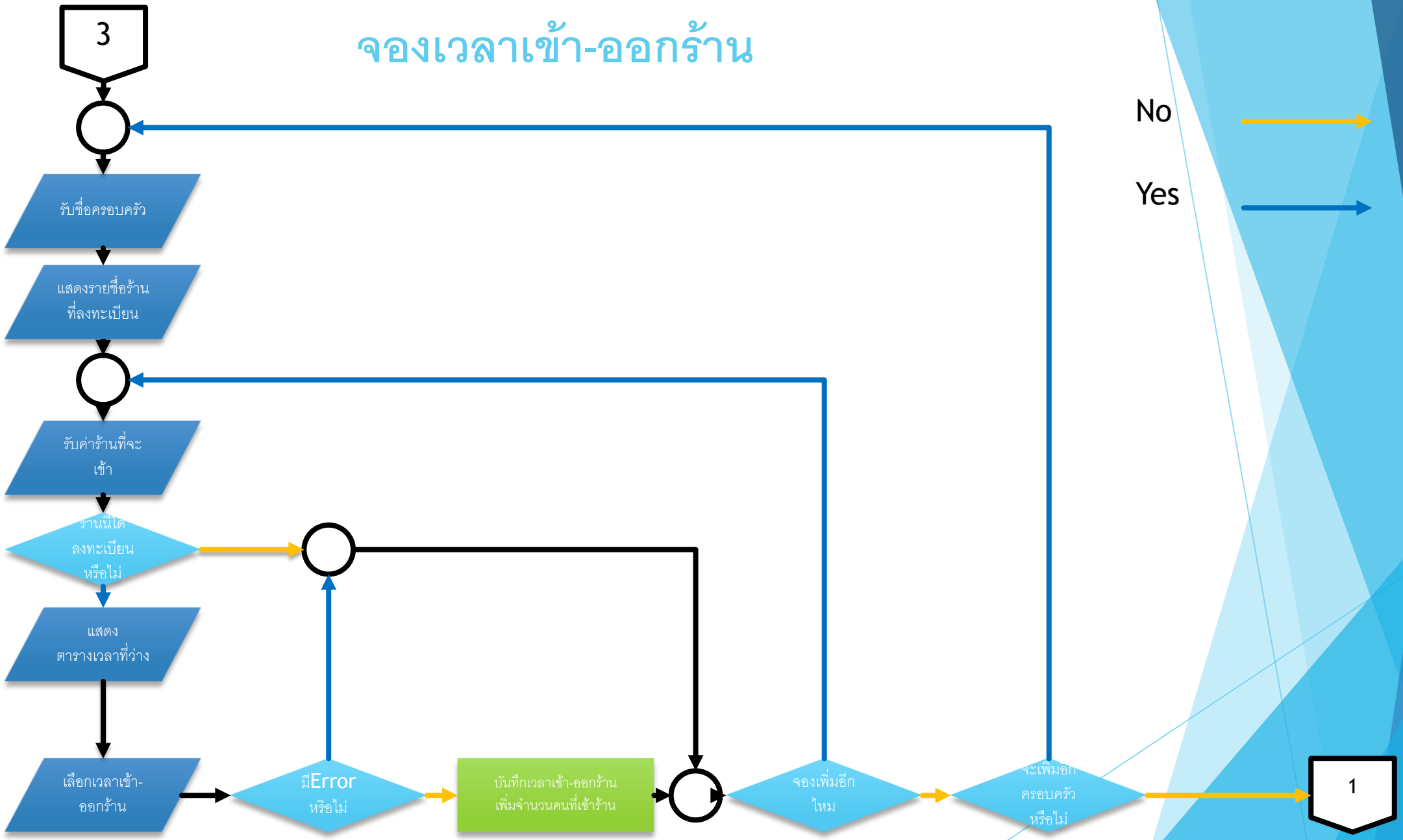
Yes

ผิดรูปแบบการออกแบบ Flow chart

ลงทะเบียนร้าน



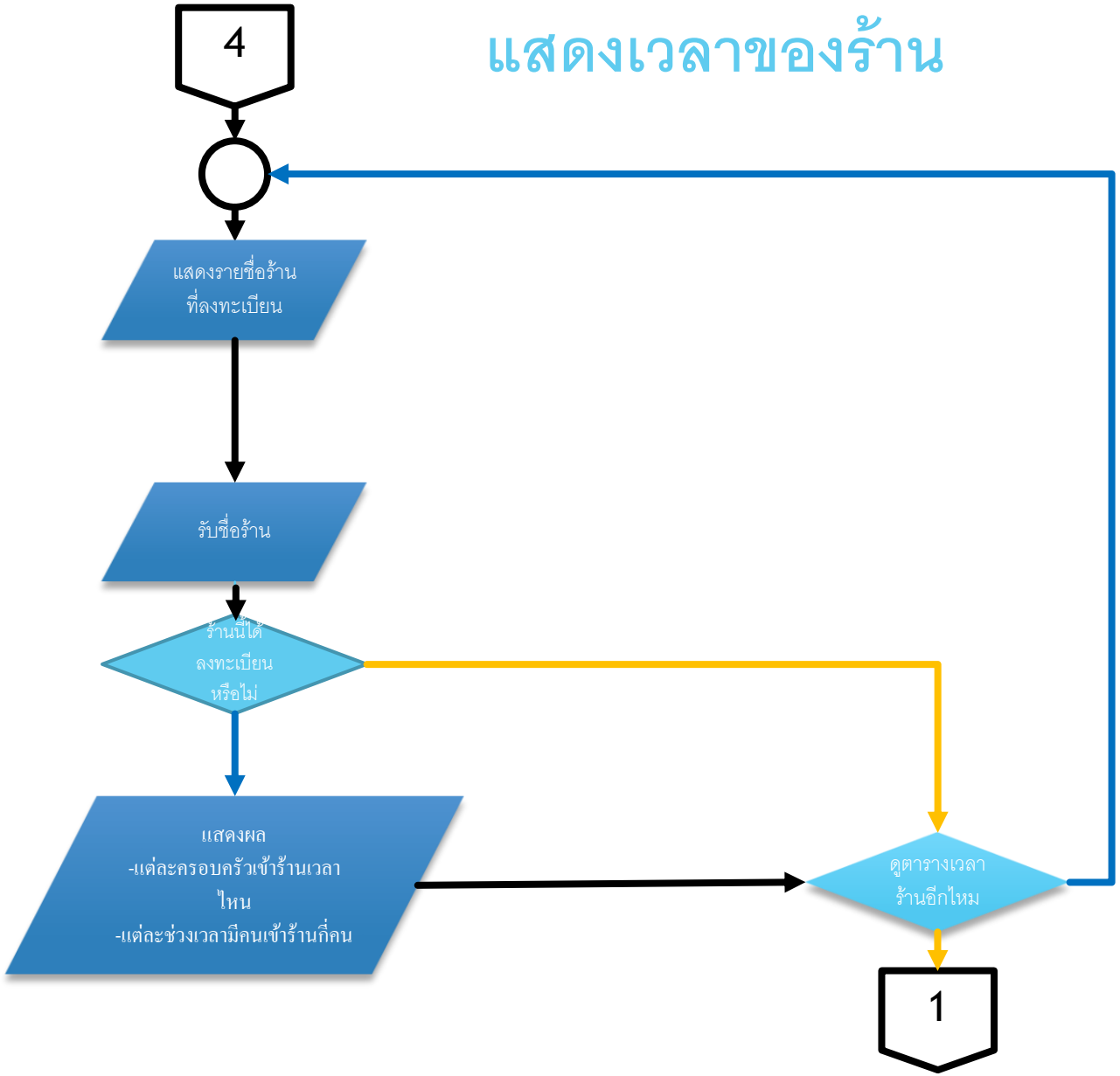
จองเวลาเข้า-ออกร้าน



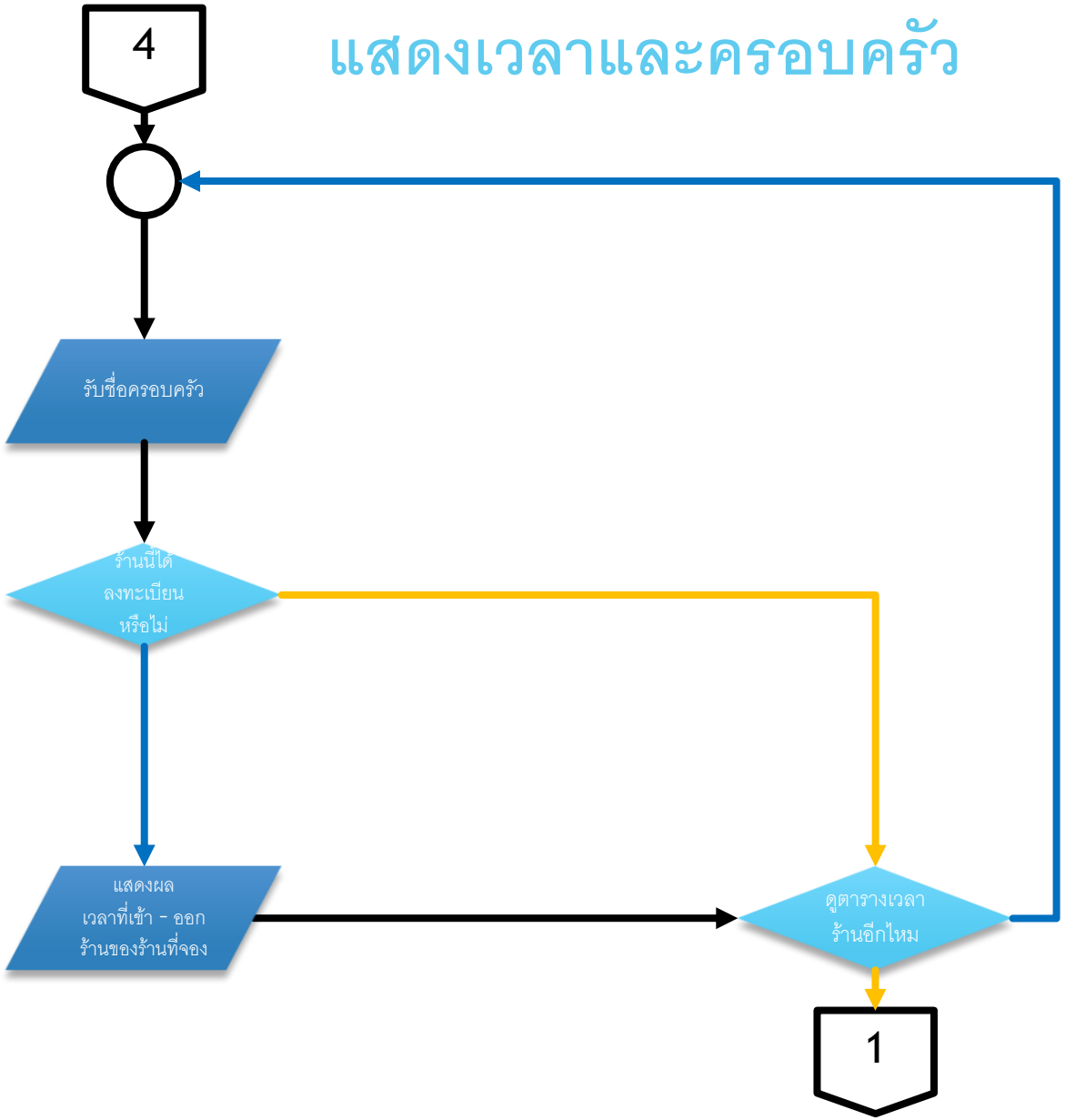
Error

- เวลาเข้าร้านมากกว่าเวลาออกร้าน
- เวลาที่จองไม่ใช้ .00 กับ .30
- เวลาทีจองเต็ม
- รับค่าผิด
- ครอบครัวนี้ได้จองเวลานี้ไปแล้วก่อนหน้านี้

แสดงเวลาของร้าน



แสดงเวลาและครอบครัว



No

Yes

REQUIREMENT

1. LINKED LIST

สร้าง linked-list town เพื่อเก็บ node family

ใน class template

```
class  family{
        int time[36];
        string name;
        family* next;
    public:
        family(string);
        void book(int,int);
        void display();
        string getname(){
            return name;
        }
        int gettime(int i){
            return time[i];
        }
        void show_node();
        void insert(family*&);
        family* move_next();
        ~family();
};
```

```
class town{
    private:
        int size;
        family*head;
    public:
        void add_family(family*&);
        void show_all();
        family* find_fam(string);
        int getsize(){
            return size;
        }
        town();
        ~town();
};
```

ใน main

```
family *pt;
town city;


pt=city.find_fam(fam_name);
if(pt==NULL){
    pt=new family(fam_name);
    city.add_family(pt);
}
else check=1;
```

2. Sorting Algorithm

ใช้ในการเก็บลำดับการเข้าร้านค้าของแต่ละครอบครัว โดยจะเรียงเป็นลำดับตามเวลาที่เข้าก่อนอยู่ใน vector order

```
void addorder(string nam,double in,double out)
{
    orderlist.push_back(order(nam,in,out));
    for(int i=0;i<orderlist.size();i++)
    {
        for(int j=i+1;j<orderlist.size();j++)
        {
            if(orderlist[i].returnin()>orderlist[j].returnin())
            {
                swap(orderlist[i],orderlist[j]);
            }
        }
    }
}
```

3. Class with constructor

3.1 Class family

เป็น class ของ node family เก็บ ชื่อครอบครัว ตารางเวลา และ address ของครอบครัวถัดไป

```
class  family{
        int time[36];
        string name;
        family* next;
    public:
        family(string);
        void book(int,int);
        void display();
        string getname(){
            return name;
        }
        int gettime(int i){
            return time[i];
        }
        void show_node();
        void insert(family*&);
        family* move_next();
        ~family();
};
```

```
family::family(string s){
            name=s;
            for(int i=0;i<36;i++){
                time[i]=0;
            }
            next=NULL;
}
```

3.2 Class town

เป็น linked list ที่เก็บ node family

```
class town{
    private:
        int size;
        family*head;
    public:
        void add_family(family*&);
        void show_all();
        family* find_fam(string);
        int getsize(){
            return size;
        }
        town();
        ~town();
};
```

```
town::town(){
    size=0;
    head=NULL;
}
```

3.3 Class building

เป็น class ของร้านค้าต่าง ๆ ภายในเมือง

```
class building{
private:
    int time[36];
    string name;
    double open;
    double close;
    int people;
    vector<order> orderlist;
    vector<int> gotoshop;
public:
    building(string);
    building(string,int);
    building(string,int,double,double);
    ~building();
```

```
building::building(string nam,int peo,double ope,double clo)
{
    name=nam;
    people=peo;
    open=ope;
    close=clo;
    int op=open*2-8;
    system("CLS");
    cout<<"Shop "<<name<<" register successful\n";
    if(open!=(int)open)
    {
        op++;
    }
    int cl=close*2-8;
    if(close!=(int)close)
    {
        cl++;
    }
    for(int i=0;i<36;i++)
    {
        if(i>=op&&i<cl)
            time[i]=0;
        else
            time[i]=people;
    }
    shownode();
}
```

3.4 Class order

เป็น class ของครอบครัวที่ลงทะเบียนในแต่ละร้านค้า

```
class order
{
private:
    string famname;
    double timein;
    double timeout;
public:
    order(string name,double in,double out)
    {
        famname=name;
        timein=in;
        timeout=out;
    }
    void shownode()
    {
        //cout<<fixed();
        cout <<setprecision(2)<<fixed;
        cout<<famname<<" family go to shop between "<<timein<<" and "<<timeout<<endl;
    }
    double returnin()
    {
        return timein;
    }
};
```

4. Polymorphism

4.1 ใช้ overloading constructor ในการประกาศ class building ทั้ง 3 รูปแบบ

```
building::building(string nam,int peo,double ope,double clo)
{
    name=nam;
    people=peo;
    open=ope;
    close=clo;
    int op=open*2-8;
    system("CLS");
    cout<<"Shop "<<name<<" register successful\n";
    if(open!=(int)open)
    {
        op++;
    }
    int cl=close*2-8;
    if(close!=(int)close)
    {
        cl++;
    }
    for(int i=0;i<36;i++)
    {
        if(i>=op&&i<cl)
            time[i]=0;
        else
            time[i]=people;
    }
    shownode();
}
```

```
building::building(string nam,int peo)
{
    system("CLS");
    name=nam;
    people=peo;
    open=6.00;
    close=18.00;
    int op=open*2-8;
    cout<<"Shop "<<name<<" register successful\n";
    if(open!=(int)open)
    {
        op++;
    }
    int cl=close*2-8;
    if(close!=(int)close)
    {
        cl++;
    }
    for(int i=0;i<36;i++)
    {
        if(i>=op&&i<cl)
            time[i]=0;
        else
            time[i]=people;
    }
    shownode();
}
```

```
building::building(string nam)
{
    system("CLS");
    name=nam;
    people=1;
    open=6.00;
    close=18.00;
    int op=open*2-8;
    cout<<"Shop "<<name<<" register successful\n";
    if(open!=(int)open)
    {
        op++;
    }
    int cl=close*2-8;
    if(close!=(int)close)
    {
        cl++;
    }
    for(int i=0;i<36;i++)
    {
        if(i>=op&&i<cl)
            time[i]=0;
        else
            time[i]=people;
    }
    shownode();
}
```

4.2 ใช้ overloading operator ในการจองเวลาของแต่ละร้านค้าใน class building

```
void operator+(int x)
{
    ++time[x];
}

vector<building> LL;

for(int j=entt;j<outt;j++)
{
     LL[i]+j;
     pt->book(j,i+1);

}
```

5. Exception Handling

ใช้ในการดัก input ที่ไม่ถูกต้องทั้งหมด เช่น เป็นตัวอักษร เป็นจำนวนลบ หรือเวลาปิดมาก่อนเวลาเปิด เป็นต้น

```
try
{
    avilableshop(LL[tt]);
    cout<<"Input time when you will enter: ";
    ent_h=100;
    ent_mn=100;
    out_h=100;
    out_mn=100;
    if(!scanf("%d.%d",&ent_h,&ent_mn))
    {
        throw 1;
    }
    if(ent_h>24||ent_h<0||ent_mn<0||ent_mn>=60)
        throw 1;
    if(ent_h>22||ent_h<4||ent_mn<0||ent_mn>=60)
        throw 2;
    if(ent_h==22&&ent_mn!=0)
        throw 2;
    if(ent_mn!=30&&ent_mn!=0)
        throw 4;
    cout<<"Input time when you will leave: ";
    if(!scanf("%d.%d",&out_h,&out_mn))
    {
        throw 1;
    }
```

```
if(out_h>24||out_h<0||out_mn<0||out_mn>=60)
    throw 1;
if(out_h>22||out_h<4||out_mn<0||out_mn>=60)
    throw 2;
if(out_h==22&&out_mn!=0)
    throw 2;
if(out_mn!=30&&out_mn!=0)
    throw 4;
if((out_h*60)+out_mn<=(ent_h*60)+ent_mn)
    throw 3;
ent = (double)ent_h+(double)(ent_mn*0.01);
out = (double)out_h+(double)(out_mn*0.01);
entt=ent*2-8;
if(ent!=(int)ent)
    entt++;
outt=out*2-8;
if(out!=(int)out)
    outt++;
for(i=entt;i<outt;i++){
    if(pt->gettime(i)!=0){
        throw 5;
        break;
    }
}

if (res.compare(LL[i].getname())==0)
{
    for(int j=entt;j<outt;j++)
    {
        if(LL[i].gettime(j)>=LL[i].getpeople())
            throw 0;
    }
```

```
catch(int err)
{
    if(err==0)
    {
        cout<<"This time is full\n";
        cout<<"Press F for select shop again, any key for add time again\n     ";
        returntime=_getch();
        if(returntime=='F'||returntime=='f')
        {
            cin.clear();
            cin.ignore(5000,'\n');
            system ("CLS");
            break;
        }
        //cin.clear();
          //  cin.ignore(5000,'\n');
            system ("CLS");
    }


 else if(err==1)
 {
     cout<<"Invalid input\n";
     cout<<"Press F for select shop again, any key for add time again\n     ";
     returntime=_getch();
     if(returntime=='F'||returntime=='f')
     {
         cin.clear();
         cin.ignore(5000,'\n');
         system ("CLS");
         break;
     }
   //  cin.clear();
       // cin.ignore(5000,'\n');
         system ("CLS");
 }
```

Limitation

1. ไม่สามารถยกเลิกการจองได้

2. ตารางเวลาในการจองเป็นช่องละ 30 นาที ซึ่งอาจไม่ละเอียดเพียงพอ

3. ไม่มีระบบป้องกันหรือยืนยันตัวตน ทำให้การจองไม่มีความปลอดภัย อาจมีคนอื่นมาใช้สิทธิเราได้ หรือ จองเล่นจนร้านค้าเต็ม

4. รับ Input ได้มากสุด 5000 ตัว

5. เวลาเคอร์ฟิวไม่สามารถเปลี่ยนแปลงได้

6. ไม่ได้จำกัดว่าครอบครัวหนึ่งสามารถจองได้กี่ร้านค้าและร้านค้าละกี่ชั่วโมง

7. ไม่มีระบบล้างตารางเวลาทั้งหมด หากต้องการ Reset ต้องปิดโปรแกรมแล้วเปิดใหม่