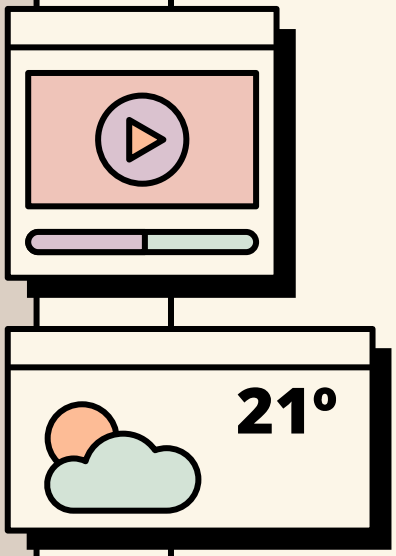
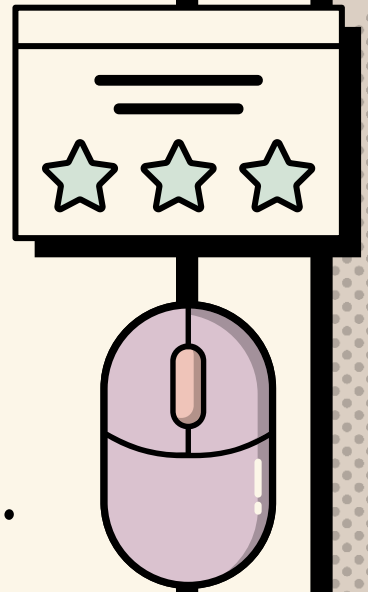
ICS212 Final Group Project





By: Landon, Jaymond, Kimberly, and Stephanie

Group Distribution

Kimberly	Checkin/Checkout function, Customer Class,
Landon	Checkin/Checkout function, RoomBill function, Troubleshooting
Jaymond	searchCustomer function, main/menu, calculateStayDuration function
Stephanie	Room class, addRoom func, view available rooms func, guest summary func

01

Classes

Classes :)

Room Class

Setters

setRoomNum(int), setAC(char), setComfortType(char), setBedSize(char), setRate(int), setBedNum(int), setOccupied(), setVacant()

Getters

getRoomNum(), getAC(), getComfortType(), getBedSize(), getRate(), getBedNum(), getStatus()

Methods

N/A

Member Variables

```
38 class Room {
39     private:
40         int roomNum;
41         char AC; // 'Y' if it has AC, 'N' if no AC
42         char comfortType; // 'S' if it is a suite, 'N' if not a suite
43         char bedSize; // 'F' if full, 'Q' if queen, 'K' if king
44         double rate; //$ per daily
45         int bedNum;//number of beds in the room
46         char status = 'V'; //'O' if occupied, 'V' if vacant.
47         //Is initialized to 'V' because rooms are automatically not occupied when added.
48
```

Setters & Getters

```
49      public:
50          void setRoomNum(int num) {
51              roomNum = num;
52              return;
53          }
54
55          void setAC(char ac) {
56              AC = ac;
57              return;
58          }
59
60          void setComfortType(char sn) {
61              comfortType = sn;
62              return;
63          }
64
65          void setBedSize(char fqk) {
66              bedSize = fqk;
67              return;
68          }
69
70          void setRate(double num) {
71              rate = num;
72              return;
73          }
74
75          void setBedNum(int num) {
76              bedNum = num;
77              return;
78          }
79
80          void setOccupied() {
81              status = 'O';
82              return;
83          }
84
85          void setVacant() {
86              status = 'V';
87              return;
88          }
```

```
90          int getRoomNum() {
91              return roomNum;
92          }
93
94          char getAC() {
95              return AC;
96          }
97
98          char getComfortType() {
99              return comfortType;
100         }
101
102         char getBedSize() {
103             return bedSize;
104         }
105
106         double getRate() {
107             return rate;
108         }
109
110         int getBedNum() {
111             return bedNum;
112         }
113
114         char getStatus() {
115             return status;
116         }
117 };
```

Customer Class

Setters

setFName(), setLName(),
setAddress(), setPayment(),
setRoomNumber(int x),
setPhone(), setId(),
setCheckIn(), setCheckOut()

Getters

getFName(), getLName(),
getAddress(), getPayment(),
getRoomNumber(), getInMonth(),
getInDay(), getInYear(),
getOutMonth(), getOutDay(),
getOutYear(), getPhone(), getId(),
getCheckIn(), getCheckOut()

Methods

checkDate()
checkPhone()

Member Variables

```
class Customer{
    private:
    std::string firstName;
    std::string lastName;
    std::string address;
    double phone;
    std::string id;
    std::string checkIn;
    int inMonth;
    int inDay;
    int inYear;
    int outMonth;
    int outDay;
    int outYear;
    std::string checkOut;
    int payment;
    int roomNum;
```

Setters

```
//kim
void setRoomNumber(int x){
    roomNum = x;
    return;
}
```

```
void setFName(){
    std::getline(std::cin, firstName);
}
void setLName(){
    std::getline(std::cin, lastName);
}
void setAddress(){
    std::getline(std::cin, address);
}
```

```
void setPayment(){
    int p;
    std::cin >> p;
    payment = p;
}
```

```
void setId(){
    std::getline(std::cin, id);
}
//e.g. 04/28/2023, gets date string and finds '/' to parse and convert the date to int after
void setCheckIn(){
std::string c;
while(1){
    std::getline(std::cin, c);
    if(!checkDate(c)){
        std::cout << "Invalid format. Please try again: ";
    } else break;
    }
    checkIn = c;
    std::stringstream stream(c);
    string s;
    std::getline(stream, s, '/');
    inMonth = stoi(s);
    std::getline(stream, s, '/');
    inDay = stoi(s);
    std::getline(stream, s, '/');
    inYear = stoi(s);
}
```

```
//e.g. 1234567890 -> 10
void setPhone(){
    std::string p;
    while(true){
        std::getline(std::cin, p);
        if(checkPhone(p)){
            phone = std::stod(p);
            break;
        } else std::cout << "Invalid phone number. Please try again: ";
    }
}
void setId(){
    std::getline(std::cin, id);
}
```

```
void setCheckOut(){
    std::string c;
    while(1){
    std::getline(std::cin, c);
    if(!checkDate(c)){
        std::cout << "Invalid. Please try again: ";
    } else break;
    }
    checkOut = c;
    std::stringstream stream(c);
    string s;
    std::getline(stream, s, '/');
    outMonth = stoi(s);
    std::getline(stream, s, '/');
    outDay = stoi(s);
    std::getline(stream, s, '/');
    outYear = stoi(s);
}
```

Getters

```
int getRoomNumber(){
   return roomNum;
}
int getInMonth(){
  return inMonth;
}
int getInDay(){
    return inDay;
}
int getInYear(){
    return inYear;
}
int getOutMonth(){
    return outMonth;
}
int getOutDay(){
    return outDay;
}
int getOutYear(){
    return outYear;
}
```

```
std::string getFName(){
    return firstName;
}
std::string getLName(){
    return lastName;
}
std::string getAddress(){
    return address;
}
double getPhone(){
    return phone;
}
std::string getId(){
    return id;
}
std::string getCheckIn(){
    return checkIn;
}
std::string getCheckOut(){
    return checkOut;
}
int getPayment(){
    return payment;
}
```

Validations

```cpp
bool checkDate(const std::string& date){
    if(date.length() != 10){
        return false;
    }
    for(int i = 0; i < 10; i++){
        if(i == 2 || i == 5){
            if(date[i] != '/'){
                return false;
            }
        }
        else if(!isdigit(date[i])){
            return false;
        }
    }
    return true;
}
```

```cpp
bool checkPhone(const std::string& phone){
    if(phone.length() != 10){
        return false;
    } else return true;
}
```

02

Functions

Add room, Display available rooms, Check in/out, Search Customer, Calculate bill, Calculate stay duration

Add Room

```
122 void addRoom(Room &theRoom) {
123     cout << "Enter room number: ";
124     int roomnum;
125     cin >> roomnum;
126     while (roomnum < 1) { //checks if user entered a number less than 1
127         cout << "Error: Room number can't be less than or equal to 0. Enter room number: ";
128         cin >> roomnum;
129     }
130     theRoom.setRoomNum(roomnum);
131 
132     cout << "Room with AC/No AC (Y/N): ";
133     char ac;
134     cin >> ac;
135     while ( (ac != 'Y') && (ac != 'N')) { //checks if user entered 'Y' or 'N'
136         cout << "Error: User must input 'Y' or 'N'. Room with AC/No AC (Y/N): ";
137         cin >> ac;
138     }
139     theRoom.setAC(ac);
140 
141     cout << "Type of comfort (S/N): ";
142     char sn;
143     cin >> sn;
144     while ( (sn != 'S') && (sn != 'N')) { //checks if user entered 'S' or 'N'
145         cout << "Error: User must input 'S' or 'N'. Type of comfort (S/N): ";
146         cin >> sn;
147     }
148     theRoom.setComfortType(sn);
```

```
150     cout << "Bed Size (F/Q/K): ";
151     char fqk;
152     cin >> fqk; //checks if user input 'F', 'Q', or 'K'
153     while ( (fqk != 'F') && (fqk != 'Q') && (fqk != 'K')) {
154         cout << "Error: User must input 'F', 'Q', or 'K'. Bed size (F/Q/K): ";
155         cin >> fqk;
156     }
157     theRoom.setBedSize(fqk);
158 
159     cout << "Number of beds: ";
160     int bednum;
161     cin >> bednum;
162     while (bednum < 1) { //checks if number of beds is at least 1
163         cout << "Error: Number of beds must be at least 1. Number of beds: ";
164         cin >> bednum;
165     }
166     theRoom.setBedNum(bednum);
167 
168     cout << "Daily Rate ($): ";
169     double dailyRate;
170     cin >> dailyRate;
171     while (dailyRate <= 0) { //checks if daily rate is greater than 0
172         cout << "Error: Daily rate must be greater than 0. Daily rate ($): ";
173         cin >> dailyRate;
174     }
175     theRoom.setRate(dailyRate);
176 
177 
178     cout << "\nRoom Added Successfully!\n";
179 }
```

Display Available Rooms

```
183 void displayRoom(Room theRoom) {
184     cout << "Room number: " << theRoom.getRoomNum() << "\n";
185     cout << "AC: ";
186     if (theRoom.getAC() == 'Y') {
187         cout << "Yes\n";
188     } else {
189         cout << "No\n";
190     }
191 
192     cout << "Room type: ";
193     if (theRoom.getComfortType() == 'S') {
194         cout << "Suite\n";
195     } else {
196         cout << "Not suite\n";
197     }
198     cout << "Bed size: ";
199     if (theRoom.getBedSize() == 'F') {
200         cout << "Full\n";
201     } else if (theRoom.getBedSize() == 'Q') {
202         cout << "Queen\n";
203     } else {
204         cout << "King\n";
205     }
206     cout << "Number of beds: " << theRoom.getBedNum() << "\n";
207     cout << "Daily rate($): " << fixed << setprecision(2) << theRoom.getRate() << "\n";
208 
209     return;
210 }
```

```
215 
216 void viewAvailable(Room rooms[], int roomCount) {
217     cout << "---Available Rooms---\n";
218     for (int i = 0; i < roomCount; i++) {
219         if (rooms[i].getStatus() == 'V') {
220             displayRoom(rooms[i]);
221             cout << "\n";
222         }
223     }
224     return;
225 }
```

Check In

```
//kimberly
//checks in customers for the amount of customers and rooms available
void checkIn(Room r[], Customer c[], int &roomCount, int &cNum){
    int roomNumber, found = -1;
    std::cout << "Enter Room number: ";
    std::cin >> roomNumber;
    std::cin.ignore();
    //excludes negative room numbers
    if (roomNumber < 0) {
        cout << "Invalid room number.";
        return;
    }
    for (int i = 0; i < roomCount; i++) {
        if (r[i].getRoomNum() == roomNumber) {
            found = i;
            break;
        }
    }
    if (found == -1) {
        cout << "Room does not exist.";
        return;
    }
    if (r[found].getStatus() != 'V') {
        cout << "Room is already booked.";
        return;
    }
```

```
    //sets customer information
        std::cout << "Enter booking id: ";
        c[cNum].setId();
        std::cout << "Enter Customer First Name (First Name): ";
        c[cNum].setFName();
        std::cout << "Enter Customer Last Name (Last Name): ";
        c[cNum].setLName();
        std::cout << "Enter Address (city only): ";
        c[cNum].setAddress();
        std::cout << "Enter Phone: ";
        c[cNum].setPhone();
        std::cout << "Check-in Date (MM/DD/YYYY): ";
        c[cNum].setCheckIn();
        std::cout << "Check-out Date (MM/DD/YYYY): ";
        c[cNum].setCheckOut();
        std::cout << "Enter Advance Payment: ";
        c[cNum].setPayment();
        r[found].setOccupied();
        c[cNum].setRoomNumber(roomNumber);
}
```

Check Out

```
void checkOut (Customer customers[], Room rooms[],int size) {
  int i;
 int roomNum;
 cout << "Checking out customer... \n";
 //function to check out a customer
 cout << "Enter Room Number: ";
 cin >> roomNum;
 //search for the customer in the array
 for (i = 0; i < size; i++) {
     if (rooms[i].getRoomNum() == roomNum) {
         cout << "Customer found: " << customers[i].getFName() << " " << customers[i].getLName() << endl;
         break;
     }
     else {
         cout << "Customer not found." << endl;
     }
 }
 cout << "Check IN Date: " << customers[i].getCheckIn() << endl;
 cout << "Check OUT Date: " << customers[i].getCheckOut() << endl;
//call function
int stayLength = calculateStayDuration(customers[i].getInDay(), customers[i].getOutDay(), customers[i].getInMonth(),
customers[i].getOutMonth(), customers[i].getInYear(), customers[i].getOutYear());
cout << "Length of Stay: " << stayLength << endl;
 //this function will update the status of the room to vacant
 //and calculate the bill for the customer
  rooms[i].setVacant();
 cout << "Customer Last Name: " << customers[i].getLName() << endl;
 cout << "Customer First Name: " << customers[i].getFName() << endl;
 cout << "Room Number: " << rooms[i].getRoomNum() << endl;
 cout << "Address:" << customers[i].getAddress() << endl;
 cout << "Phone: " << customers[i].getPhone() << endl;
 roomBill(rooms[i].getRate(), stayLength);
 cout << "Advance Payment: " << customers[i].getPayment() << endl;
 cout << "Total Amount Due: " << (rooms[i].getRate() * stayLength) - customers[i].getPayment() << endl;
cout << "Customer checked out successfully." << endl;
cout << "Room status updated to vacant." << endl;
cout << "Thank you for staying with us!" << endl;
}
```

Search Customer

```
int searchCustomer(Customer customers[], int guests)
{
    int room = -1, count = 0, nonemp = 0;
    string last;
    bool in = false;
    char check[100];

    cout<<"Enter Customer's last name: ";

    check[0] = getc(stdin);
    in = false;
    count = 0;
    nonemp = 0;
    while (!in)
    {
        check[0] = getc(stdin);
        while (check[count] != '\n')
        {
            if (isalpha(check[count]))
            nonemp++;
            check[++count] = getc(stdin);
        }
        if (count > 0 && nonemp > 0)
        {
            check[count] = '\0';
            last = check;
            in = true;
        }
        else
        {
            std::cout << "\nCustomer name cannot be empty. Please enter a valid customer name.\n";
            std::cout << "Enter Customer's Last Name: ";
            count = 0;
        }
    }
```

```
for (int i = 0; i < guests; i++)
{
    if (customers[i].getLName() == last)
    {
        room = i;
    }
}

if (room > -1)
{
    int inMonth = stoi(customers[room].getCheckIn().substr(0,2)), outMonth = stoi(customers[room].getCheckOut().substr(0,2));
    int inYear = stoi(customers[room].getCheckIn().substr(6)), outYear = stoi(customers[room].getCheckOut().substr(6));
    int inDay = stoi(customers[room].getCheckIn().substr(3,2)), outDay = stoi(customers[room].getCheckOut().substr(3,2));

    cout<<"\nMonth: "<<inMonth<<", Day: "<<inDay<<", Year: "<<inYear
    <<"\nMonth: "<<outMonth<<", Day: "<<outDay<<", Year: "<<outYear
    <<"\nNumber of Days during the stay: "<<calculateStayDuration(inDay, outDay, inMonth, outMonth, inYear, outYear)
    <<"\nCustomer First Name : "<<customers[room].getFName()
    <<"\nCustomer Last Name : "<<customers[room].getLName()
    <<"\nAddress (city only) : "<<customers[room].getAddress()
    <<"\nPhone : "<<customers[room].getPhone()
    <<"\nArrival Date : "<<customers[room].getCheckIn()
    <<"\nDeparture Date : "<<customers[room].getCheckOut()
    <<"\nLength of Stay (in days) : "<<calculateStayDuration(inDay, outDay, inMonth, outMonth, inYear, outYear)
    <<"\nBooking ID : "<<customers[room].getId()
    <<"\nRoom Number : "<<customers[room].getRoomNumber();
}
return 0;
```

Room Bill Calculator

```
576 //Landon
577 //separate function to calculate room bill
578 void roomBill(double dailyRate, int daysStayed) {
579  double totalBill;
580 totalBill = dailyRate * daysStayed;
581 cout << "$" << fixed << setprecision(2) << totalBill << endl;
582  //function to calculate the bill for a room
583 }
```

Stay Duration

```
int calculateStayDuration(int inDay, int outDay, int inMonth, int outMonth, int inYear, int outYear)
{
    int dayTotal = 0;
```

```
while (outYear > inYear) {
    if (inDay > 0) {
        if (inMonth == 4 || inMonth == 6 || inMonth == 9 || inMonth == 11) {
            dayTotal += 30 - inDay;
        } else if (inMonth == 2) {
            if (inYear % 4 == 0 && (inYear % 100 > 0 || inYear % 400 == 0)) {
                dayTotal += 29 - inDay;
            } else
                dayTotal += 28 - inDay;
        } else
            dayTotal += 31;
        inDay = 0;
    }

    if (inMonth == 4 || inMonth == 6 || inMonth == 9 || inMonth == 11) {
        dayTotal += 30;
    } else if (inMonth == 2) {
        if (inYear % 4 == 0 && (inYear % 100 > 0 || inYear % 400 == 0)) {
            dayTotal += 29;
        } else {
            dayTotal += 28;
        }
    } else
        dayTotal += 31;

    inMonth++;
    if (inMonth == 13) {
        inYear++;
        inMonth = 1;
    }
}
```

```
while (inMonth < outMonth) {
    if (inDay > 0) {
        if (inMonth == 4 || inMonth == 6 || inMonth == 9 || inMonth == 11) {
            dayTotal += 30 - inDay;
        } else if (inMonth == 2) {
            if (inYear % 4 == 0 && (inYear % 100 > 0 || inYear % 400 == 0)) {
                dayTotal += 29 - inDay;
            } else {
                dayTotal += 28 - inDay;
            }
        } else
            dayTotal += 31 - inDay;
        inDay = 0;
    }

    if (inMonth == 4 || inMonth == 6 || inMonth == 9 || inMonth == 11) {
        dayTotal += 30;
    } else if (inMonth == 2) {
        if (inYear % 4 == 0 && (inYear % 100 > 0 || inYear % 400 == 0)) {
            dayTotal += 29;
        } else
            dayTotal += 28;
    } else
        dayTotal += 31;
    inMonth++;
}
```

```
if (inDay == 0)
        inDay++;

while (inDay < outDay)
{
    dayTotal++;
    inDay++;
}

return ++dayTotal;
```

Guest Summary

```
void guestSummary(Customer customers[], int customerCount) { //display summary of all current guest
    for (int i = 0; i < customerCount; i++) {
        cout << "Customer First Name: " << customers[i].getFName() << "\n";
        cout << "Customer Last Name: " << customers[i].getLName() << "\n";
        cout << "Address (city only): " << customers[i].getAddress() << "\n";
        cout.precision(0);
        cout << fixed << "Phone: " << customers[i].getPhone() << "\n";
        cout << "Arrival Date: " << customers[i].getCheckIn() << "\n";
        cout << "Departure Date : " << customers[i].getCheckOut() << "\n";
        cout << "Length of Stay (in days): " << calculateStayDuration(customers[i].getInDay(), customers[i].getOutDay(),
         customers[i].getInMonth(), customers[i].getOutMonth(), customers[i].getInYear(), customers[i].getOutYear());
        cout << "\n\n";
    }

    return;
}
```

Main Menu

```
int main()
{
    int opt = 0, roomCount = 0, guests, guestCount = 0;
    Room rooms[100];
    Customer customers[100];

    do
    {
        cout << "---Hotel Management System---\n"
        <<"1. Manage Rooms\n"
        <<"2. Check-In Room\n"
        <<"3. Available Rooms\n"
        <<"4. Search Customer\n"
        <<"5. Check-Out Room\n"
        <<"6. Guest Summary Report\n"
        <<"7. Exit\n"
        <<"\nEnter Option: ";
        cin>>opt;
```

```
        switch(opt)
        {
            case 1:
                addRoom(rooms[roomCount]);
                roomCount++;
                std::cout << std::endl << std::endl; //added for formatting -kim
                break;
            case 2:
                //kim
                checkIn(rooms, customers, roomCount, guestCount);
                guestCount++;
                break;
            case 3:
                viewAvailable(rooms, roomCount);
                break;
            case 4:
                searchCustomer(customers, guestCount);
                break;
            case 5:
            checkOut(customers, rooms, guestCount);
                break;
            case 6:
            guestSummary(customers, guestCount);
                break;
            case 7:
                cout<<"Thank you for using our Hotel Management System!";
                break;
            default:
                cout<<"Invalid option. Please choose one of the displayed options.";
                break;
        }
    } while(opt != 7);
    return 0;
}
```

03

Pros/Cons

Pro vs Cons about our process and coding style

Pros

- No arguments (yay!)
- Everyone was constantly working, no one was without something to do
- All the code fits together somehow even though we all have different coding styles
- Helped team members when needed

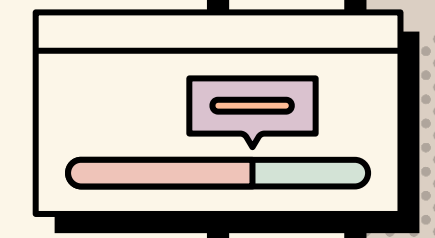
Cons

- Communication Issues (2 people worked on the same function without anyone knowing)
- Time Management (ran out of time before we could fully test our code)

How to improve?

- Better communication (perhaps write down exactly what everyone is working on)
- Better time management (keep an eye on the clock. If someone is taking a long time, help them out faster)

Click



Thanks!

Does anyone have any questions?