**Chermaine Cheang - Final Project Design**

**<u>Theme</u>**

Your friends have been captured by an evil force. To bring them back, you have to look for their name tags hidden somewhere in the building, and place the name tags on the back of their body to lift the evil magic confining them before the time is up.

**<u>Scenario</u>**

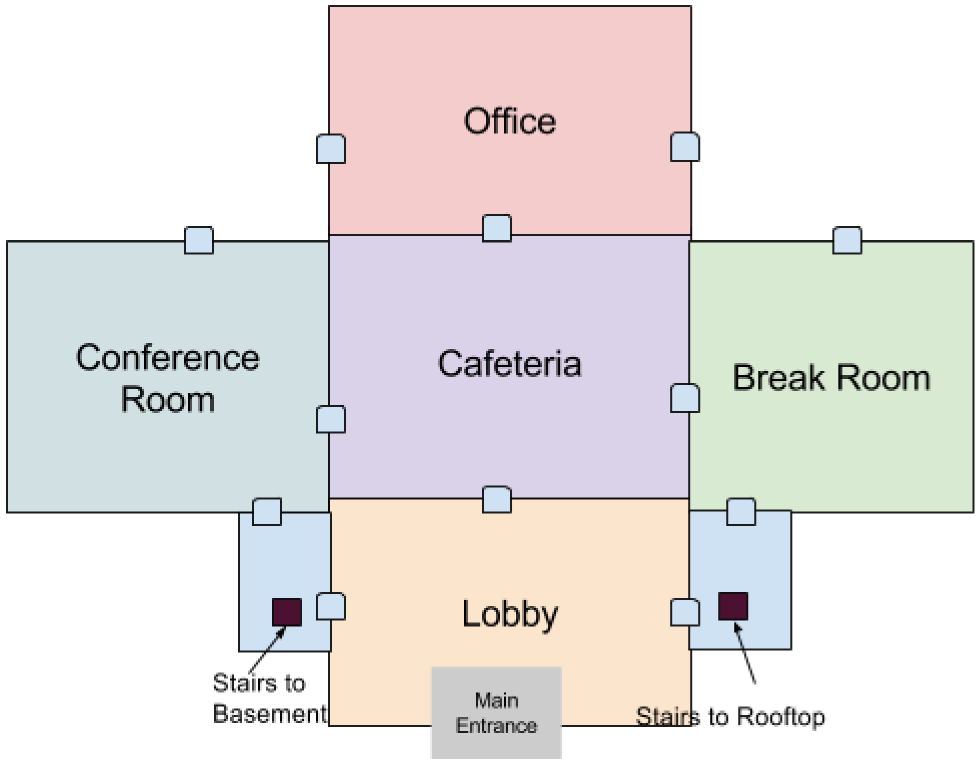
Your team has decided to stay overnight working on a project in the office. Halfway through the night, you were very tired and you fell asleep in the breakroom. When you woke up the next morning, you realized all your team members are gone. They are no way to be found. You tried calling them, but realized they left their phones and keys in the office. You looked around and found a note on one of your co-worker desk.

> "Hahahaha!!! I have captured all your team members and put a curse on them! Save your friends by rescuing them from where they are being confined before time is up. To save your friends, look for their name tags and place it on their back"

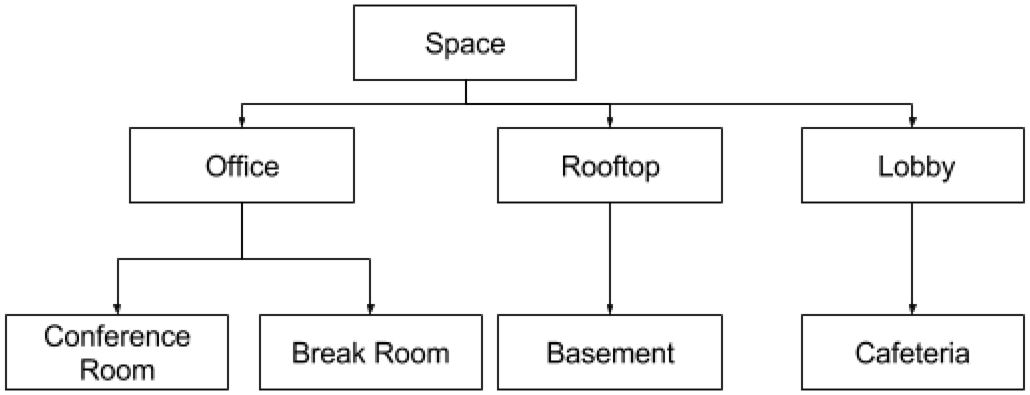
After displaying this message, the game will start.

**<u>Space</u>**

These will be rooms that I will implement for this game.



The hierarchy of my class will be as follow.



I will have an abstract space class that will have the basic information for all the other spaces. I will based on my hierarchy on how many valid doors each space will have. Office, Conference Room and Break Room each has 3 valid doors. Lobby and cafeteria have a total of 5 doors each. Basement and rooftop are only accessible by using the stairs, and has only 1 doors.

**Container for objects in each space**

I will use a queue container for storing objects in room. So every time the player enters a room and start interacting with the objects in the room, the first object in the queue will be remove for user interaction. After the user is done interacting with the object, the object will be able back to the queue. Each room will have their own objects. I will have a class for object, in which an object can be created with information they need. Then the queue container can hold an object, instead of only keeping a name of what the object is.

```
class Object
{
        Private:
                String name
                Space* location
                String description
                Bool found
                String nextDescription
        Public:
                setName
                getName
                setLocation
                getLocation
                setDescription
                getDescription
                setNextDescription
                getNextDescription
                getFound
    }
```

The variable location is used to keep track of where this object is located, while the description is used to describe the object when the player is interacting with it. In each room, I will create the objects I need and will put them in the list container. Although some room might have the same object, I do not intent to reuse the same object because their description will be different in each room there are in.

A list of objects I need for each room

- Office: desks, bookshelf
- Lobby: reception desk, drawer, key
- Cafeteria: Dining table, counter
- Kitchen: refrigerator, key, door
- Breakroom: Pool, couches, TV
- Conference room: conference table, papers
- Basement: light

To create each object and put them into their respectively link, when a constructor for a space is called, the constructor will also call a function inside Space for creating the objects in that space.
Since each space has their own objects, this function will be a pure virtual function.

```
void createObject()
{
    Create object1
    Add object1 to Queue
    Create object2
    Add object2 to Queue
    …
}
```

**<u>Container for name tags</u>**

For the name tags that the player is supposed to look for, I will have it store in a stack container. The order of the name tag being put in the stack will be random. So the first name enter the stack will be found last. The names I will use for this game will be Jon, Brandon and Rickon. At the start of the game, I will call the function to randomly place this names in order.

```
int* randomName()
{
        Int temp* = NULL;
        seed(time(0));
        Int name[3]
        Name[0] = rand()%3 + 1;
        Name[1] = rand()%3 + 1;
        while (name[0] == name[1])
                name[1] = rand()%3 + 1
        if ((name[0] == 1 && name[1] == 2) || (name[0] == 2 && name[1] == 1))
                Name[2] = 3;
        Else if ((name[0] == 1 && name[1] == 3) || (name[0]==3 && name[1]==1))
                Name[2] = 2
        Else
                Name[2] = 3

        Temp = name;
        Return temp;
}
```

Once the order of the name tag is created, the same order will be used for the body. The body will be treated as an object, and will be added to their respective spaces.

**<u>Abstract Class Space</u>**

A space abstract class that will have a special pure virtual function for special action, that is to set objects in each space

- Data members
    - Four pointers that link to other spaces, this will be stored in an array
    - Variable to indicate which floor this room is at
    - Variable for name of space to keep track of which room the player is in
    - Container to hold objects in a room - queue
    - Container to hold name tag - stack (static)
- Functions
    - Accessor and mutator functions for the array
        - Doors = new Space*[4];
    - Accessor and mutator functions for variable to indicate which floor

- Pure virtual function

```
class Space
{
	protected:
		Space* doors;
		string nameOfSpace;
		int floorNum;
		queue<Object> objectInRoom;
	public:
		Accessor and mutator functions for above variable
		function to set list in each room;
		function to get list in each room;
		Pure virtual function to create objects in each room
		Default constructor
}
```

**Character for the game**

To implement the character for this game, I will have a character class which will have all the basic information of the player. User will be prompted to give a name for the character. A variable will be used to keep track of where the player is throughout the game. This variable will be set every time the player move from one room to another. A vector container will be provided for player to store items collected throughout the game. However, there will be a limit as to how many items can be added to the vector. For the purpose of this game, I will limit the vector container to hold a maximum of 5 items. That is if vector's size() return 5, then no additional item can be added to the vector.

```
class Character
{
	Private:
		String name;
		String currentLocation;
		vector<string> itemInPocket;
		String person[3];
		Stack<string> nameTag;
	Public:
		Accessor and mutator function for data members
		setPerson will receive a pointer to an array parameter
		setNameTag  will call randName() and createName()
```

```
                        randName to randomly generate order of name
                        createName to add name to stack
        }
```

**<u>Information flow</u>**

To start the game, I will first display a welcome message for the game.

*Welcome to Save Your Friends!*

Then, I will ask if the user wants to start the game.

*Start the game?*
*Y or N*

If the user enter 'Y', then the game will start. If the user enter 'N', then the program will terminate. Once the user enter 'Y', I will create all the spaces I need for the game and create a character which will be the player, and display the objective of the game.

*Your team (You, Jon, Brandon and Sansa) has decided to stay overnight working on a project in the office. Halfway through the night, you were very tired and you fell asleep in*
*the breakroom. When you woke up the next morning, you realized all your team members are gone. They are no way to be found. You tried calling them, but realized they left their phones and keys in the office. You looked around and found a note on one of your co-worker desk.*

*"Hahahaha!!! I have captured all your team members and put a curse on them! Save your friends by rescuing them from where they are being confined before time is up. To save your friends, look for their name tags and place it on their back"*

After displaying the objective of the game, I will provide a menu for the player asking if they want to know the solution for the game. This is mainly for the grader.

*Do you want to know the solution?*
*Y or N*

If the user select 'Y', then I will let the user knows where the name tags are located, and where to find the victims in the game.

*Jon's name tag is in the office, while Jon is in the basement.*

*Etc.*

After displaying the solution to the game, the game will officially start by displaying the player's current position, and letting the player knows the first object that they can interact with. The game will start in the office.

*You are currently in the office.*
*Right in front of you is the ______. Do you want to search through the ______?*
*Y or N*

If player does not want to search through the object, the player is than prompt to look through the next object. If player wants to search through the object, the result for looking through the object will then be displayed. If user did not find anything relevant for the game, then the following message will be printed, and user will be prompted to move on to another object.

*You did not find anything useful.*
*Move on to the next object?*
*Y or N*

If the user find something relevant for the game, then the following message will be printed, and user will be prompted if they want to store the item in their pocket.

*You found _______.*
*Keep it in your pocket?*
*Y or N*

After finished examine a room, user will be prompted to move on to another room.

*Move to another room?*
*Y or N*

If player inputs 'N', then player will be asked if they want to look around the room again.

*Look around again?*
*Y or N*

If player inputs 'Y', then a list of rooms connected to the current room the player is in will be listed.

*You can go to*
*1 - ______*
*2 - ______*
*3 - ______*
*4 - ______*
*from this room.*

Once the player selected a room, the player will be brought to the room where he/she selected and the player will be able to interact with objects in the room.

If the player found a person, then the player will have to look through his pocket to see if he/she currently has the name tag for that person. If the player has the correct name tag, then the player successfully rescued one of his/her friend.

*Congratulation! You saved ____*

A counter will be used to keep track of how many people the player still has to save. When the counter is empty, then the player has won the game. However, if the counter is not empty, and the player has not move for more than 15s, the system will prompt the player if they want to continue with the game.

*You have been inactive for 15s, do you want to continue?*
*Y or N*

**Solution**

Location of name tags:

- Lobby
- Cafeteria
- Conference Room

Location of People:

- Basement
- Rooftop
- Kitchen

**Sequence of interaction with objects**

In the office, the player will first interact with the desk follow by the bookcase and another desk.

In the lobby, the player will first interact with the receptionist desk follow by the drawers in the desk.

In the cafeteria, the player will first interact with the dining tables, follow by found counter and the kitchen. The kitchen will not be an object. The kitchen will be another class which is added when the player enter the cafeteria. Once the player is done with the kitchen, it will be deleted.

In the kitchen, the player will first interact with the refrigerator, then a key, and back to the refrigerator, and finally a body

In the break room, the player will first interact with the pool table, follow by couches and tv stand

In the conference room, the player will interact with the conference table and piles of paper

In the basement, the player will interact with a light and another body

In the rooftop, the player will interact with a door and another body

**<u>Main Drive</u>**

```
Void startOption()
{
        Do {
                Cout << “Start the game?”
                Cout << “Y or N”
                Cin >> input
        } while (input is not valid)

Space* createSpace()
//create all the rooms necessary for the game
//return the first room the player will be in at the beginning of the game

Void solutionMenu()
//print out order of the name tag currently in the stack
//print out where the body for these names are located

bool interactWithSpace(Character &player)
//facilitate interaction of player with object in current space
```

```
//player is passed by reference so that item collected in this space will remain in scope when this function ends
//before returning, update player's location to the one player is entering next
//will return true if player saved a person in this room, else return false

Void createName(Stack& list)
//call randomName to generate order of the name tags
//add to nameTag's stack

Int main()
{
    Cout << "Welcome to Save Your Friends!";
    Get user choice from startOption();
    Int savedCounter = 3;
    Bool saved;
    If user choice is yes
        Space* currentSpace = createSpace()
        Cout << "Enter your name" << endl;
        Cin >> name
        Character player(name, currentSpace->getName());
        solutionMenu()

        While (savedCounter > 0){
            Saved = interactWithSpace(player)
            If saved is true
                Decrement savedCounter
        }

        If savedCounter == 0
            Player won the game

    Free memory allocated
}
```

**Testing plan**

General plan:

1. Test if all rooms/spaces are created and linked to each other correctly
2. Test if objects for each rooms are created without any errors
3. Test if a character can be created

4. Test if a name tag can be added into one of the space and have the character look for the name tag
5. Test if the character can add the name tag into its container
6. Test if character can remove item from its container, that is when the character saves a person
7. Test if program terminates correctly once the player saved the person
8. Test to generate a random order of names, add to Stack correctly and update solutionMenu accordingly
9. Test if player is able to look for all name tags in all location specified
10. Test if savedCounter is counted correctly
11. Test if player won the game

**Pseudocode**

```
class Space
{
    protected:
        Space* room1;
        Space* room2;
        Space* room3;
        Space* room4;
        string nameOfSpace;
        int floorNum;
        queue<Object> objectInRoom;
    public:
        Accessor and mutator functions for above variable
        function to set list in each room;
        function to get list in each room;
        Pure virtual function to create objects in each room
        Space()
}

class Office inherit Space
{
    Public:
        Constructor for Office inherit Space
        createObject()
        {
            Create desk1
            Create desk2
```

```
                Create bookcase
                Add desk1 to objectInRoom
                Add desk2 to objectInRoom
                Add bookcase to objectInRoom
            }
}

class Rooftop inherit Space
{
    Public:
            Rooftop()
            createObject()
            {
                Get name from Stack
                Create person with name
                Add person to objectInRoom
            }
}

class Basement inherit Space
{
    Public:
            Basement()
            createObject()
            {
                Create light
                Get name from Stack
                Create person with name
                Add light to Queue
                Add person to Queue
            }
}

Class Lobby inherit Space //will have one additional pointers compared to all the other classes
{
    Private:
            Space* newSpace;
    Public:
            Lobby()
```

```
                createObject()
                {
                        Create receptionistDesk
                        Create drawer1
                        Create drawer2
                        Add objects to Queue
                }
                setNewSpace
                getNewSpace
}

Class ConferenceRoom inherit Space
{
        Public:
                ConferenceRoom()
                createObject()
                {
                        Create conferenceTable
                        Create papers
                        Add objects to Queue
                }
}

Class BreakRoom inherit Space
{
        Public:
                BreakRoom()
                Create object()
                {
                        Create poolTable
                        Create couches
                        Create tv
                        Add all to Queue
                }
}

Class Cafeteria inherit Lobby //also have one additional pointer
{
        Public
```

```
                Cafeteria()
                Create object()
                {
                        Create diningTable
                        Create counter
                        Add to Queue
                }
}

Class Kitchen inherit Space
{
        Public:
                Kitchen()
                Create object
                {
                        Create refrigerator
                        Create key
                        Create a person with name from Stack
                        Add all to Queue
                }
}

int* createName(Stack &list)
{
        Int* order= randName();
        For (int i=0; i<3; i++)
                If order[i] == 1
                        list.push("Jon")
                If order[i] == 2
                        list.push("Brandon")
                If order[i] == 3
                        list.push("Sansa")
        Return order;
}

Space* createSpace()
//create all the rooms necessary for the game
//return the first room the player will be in at the beginning of the game
{
        Space* office = new Office
        Space* breakRoom = new BreakRoom
        Space* conferenceRoom = new ConferenceRoom
        Space* lobby = new Lobby
        Space* cafeteria = new Cafeteria
        Set office array of pointers to cafeteria, breakRoom, conferenceRoom
        Set breakRoom array of pointers to office, cafeteria, lobby
        Set conferenceRoom array of pointers to office, cafeteria, lobby
        Set lobby array of pointers to cafeteria, breakroom, conferenceRoom
        return office;
}

Void solutionMenu()
{
        String name[3];
        For (int i=0; i<3; i++)
        {
                Name[i] = stack.top();
                stack.pop();
        }

        printSolution(name[0], name[1], name[2])

        For (int i=3; i>3; i--)
        {
                stack.push(name[i]);
        }
}

Void printSolution(string name1,string name2, string name3)
{
        Cout << name1 << "'s name tag is in Lobby, and he/she is in Basement" << endl;
        Cout << name2 << "'s name tag is in Cafeteria, and he/she is in Rooftop" << endl;
        Cout << name3 << "'s name tag is in Conference Room, and he/she is in Kitchen" <<
        endl;
}

bool interactWithSpace(Character &player)
{
        Bool saved;
        Int next;
        Int count = 0;
        Space* current = player.getCurrentLocation();
        Print out current location (player.getCurrentLocation())

        If current location is kitchen or basement or rooftop
                Cout << “Found ” << player.getPerson(count) << endl;
                For (int i=0; i< vector.size(); i++)
                        If vector[i] == player.getPerson(count)
                                Cout << “Place” << player.getPerson(count) << “ name tag on this
                                person.” << endl
                                Cout << “You saved” << player.getPerson(count) << endl;
                                Saved = true;
                                vector.erase(i);
                        Else
                                Cout << “You do not have the name tag” << endl;
                                Saved = false;
                count++;
        else
                Queue objects = Get objects in current space

                interactWithObject(queue, player);
                Char move = moveToNextSpace();
                Next = doorsToNextSpace(player’s current location);

                setPlayerNewLocation(player, next);
                deleteSpace(current);

        Return saved;
}

void printCurrentLocation(string space)
{
        Cout << “Currently in “ << space << endl;
}


void interactWithObject(Queue list, Character &player)
```

```
{
    Bool saved;
    Object object = list.pop();
    Char choice, next;
    Do {
        choice = printObjectNAsk(object);
        If choice is yes
            interactingObj(object, Character &player)
        next = moveToNextObject();
        If next is yes
            list.push(object)
            Object = list.pop();
    } while (next is yes)
}

Char printObject(Object object)
{
    Char input;
    Do {
        Cout << "Right in front of you is the " << object.getName() << "." << endl;
        Cout << "Do you want to search through it?" <<endl;
        Cout << "Y or N" << endl;
        Cin >> input
    } while (!yesNoValidate(input);
    Return input;
}

void foundNameTag(Object object, Character &player)
{
    If object is dining table and player currently in cafeteria or
    If object is drawer and player currently in lobby or
    If object is paper and player currently in conference room
        String nameTag = stack.top()
        stack.pop();
        Cout << "Found " <<  nameTag << "'s nameTag" << endl;
        keepNameTagInBag(nameTag, player)
}

Void keepNameTagInBag(string name, Character &player)
```

```
{
        Do
        {
                Cout << “Do you want to keep ” << name << “‘s nameTag in pocket?” << endl;
                Cout << “Y or N”
                Cin >> input
        } while (!yesNoValidate(input));

        If input is yes
                Add name to player’s vector container
}

void interactingObj(Object obj, Character &player)
{
        Cout << object.getDescription << endl;
                if object’s found is true
                        foundNameTag(object, player)
                Else
                        Cout << “You did not find anything useful.”
}

char moveToNextObject()
{
        Char input;
        Do {
                Cout << “Move on to the next object?” << endl;
                Cin >> input
        } while (!yesNoValidate(input));
        Return input;
}

Char moveToNextSpace()
{
        Char input;
        Do {
                Cout << “Move to the next room?” << endl;
                Cin >> input
        } while (!yesNoValidate(input));
        Return input;
```

```
}

int doorsToNextSpace(Space* space)
{
        Cout << “This room is connected to” << endl;
        Int count =1;
        While (space’s array of pointers != NULL)
        {
                Cout << count << “ - ” << space->getPointer()->getName() << endl;
                Count++;
        }
        Get user choice and validate
        createSpace(space, input);
        Return user’s choice
}

Void createSpace(Space* space, int input)
{
        String name;
        Switch (input)
        Case 1: name = get location connected to first door
        Case 2: name = get location connected to second door
        Case 3: name = get location connected to third door
        Case 4: name = get location connected to fourth door
        Case 5: name =get location connected to fifth door

        If name is either conference room or lobby
                Create new Basement space
                Set basement pointer to conference room and lobby
                Set conference room and lobby pointers to basement

        If name is either break room or lobby
                Create new Rooftop space
                Set rooftop pointer to break room or lobby
                Set break room and lobby pointers to rooftop

        If name is cafeteria
                Create new kitchen space
                Set kitchen pointer to cafeteria
                Set cafeteria pointer to kitchen
}


Void setPlayerNewLocation(Character& player, int input)
{
        If input == 1
                player.setLocation(player.getLocation().getDoors(1));
        If input == 2
                player.setLocation(player.getLocation().getDoors(2));
        If input == 3
                player.setLocation(player.getLocation().getDoors(3));
        If input == 4
                player.setLocation(player.getLocation().getDoors(4));
}

Void deleteSpace(Space* space)
{
        If space.getName() is kichen
                Set cafeteria pointer pointing to kitchen to NULL
                Delete kitchen
}
```