Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

Objective: You will design and implement a text-based game or puzzle where the player moves through a series of rooms or compartments. Each space will be a class with (at least) four pointer variables that link to other spaces. You must have at least 5 spaces of at least 3 different types. You will have a space abstract class that will have a special pure virtual function. Each type of space will have a special action. You will have at least 3 derived classes for different types of spaces. You will have at least 3 derived classes for different types of spaces.

You must have some way to keep track of which space the player is in. The player will have a container (backpack, knitting bag, or notebook) to carry "items". The container must have some limit. One or more of these items will be required as part of the solution, such as a "key" to open the locked door. You should have a time limit to urge the player on. This does not mean a literal clock, just some way to prevent the 'game' from going on indefinitely. The player must interact with parts of the structure, and not just simply collect things.This can be throwing something at the monster, operating a light switch (or other control), opening doors, or singing to get the baby back to sleep.

Theme: Game of Life – Make it through with enough money for retirement. You will start off with a certain amount of funds before the game begins. Player will traverse through the different spaces (school, apartment, work, new house, lawyer's office, retirement community) in order to perform the necessary tasks before going to the retirement community. Player must have a minimum amount of money in order to get into the retirement community. Tasks player must perform:

- Get a degree from school
- Obtain a house (not an apartment)
- Get one promotion at work
- Obtain enough money to retire (set amount of money)

List of Rooms and games within rooms:

1. School:
   a. Tic Tac Toe against the computer to get your degree
   b. if you win you pay back your student loans asap
   c. if you lose, you owe more student loans because you had to go to school for a longer time than originally planned.
2. Apartment:
   a. Hangman or roll dice for your house type (trailer, rancher, mansion, etc.)
   b. Better you do, the less expensive the house and nicer the house
   c. Once you get a house, your apartment will be deleted and your house space will be created.
   d. Must remember to get and keep deed to house
3. Work:
   a. Combat game against boss
      i. Lose = must take a vacation to de-stress (losing money)
      ii. Win = get promotion! (gain money)
      iii. Roll a specific number = go to lawyer's office, your IP has been stolen! (lose money)

   b. Job: random chance that when you go into the work space that you will receive your salary's pay.

4. <u>New House:</u>
   a. Must present deed of house on first arrival or you have to go to lawyer's office
   b. Roll dice for number (and gender?) of kids on first arrival
   c. Must come here every x moves in order to get strength back up.
   d. More kids = more expensive.
5. <u>Lawyer's Office:</u>
   a. Much like monopoly's prison, only used when unfortunate roll occurs.
   b. Visit here and pay your fees, get deed if applicable, then you can leave.
6. <u>Retirement Community:</u>
   a. If you have enough money, game over – you win!
   b. If you do not have enough money you will be rejected from entering and must go back to work in order to get more money before the time runs out.

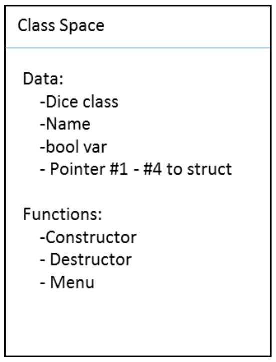
Breaking Down the Code:

Data Needed	Actions
Person Class - Money (double) - StudentLoans (double) - House Type (enum?) - Number of kids (int) - Location	Person - getStudentLoans - getMoney - move locations
School - TicTacToe Class - Degree (will be a bool based on game results)	School - Increase Student Loans (every n moves?) - Pay Student Loans - Play TicTacToe Game
Apartment - Hangman/Dice Class - House price (array of ints)	Apartment - Increase house debt - Get house debt - Play hangman game / Roll Dice
Work - Combat class - Dice class - Salary (double)	Work - Attack - Defense - Roll Dice - getSalary - Increase Salary (promotion)
New House - Deed (bool based on having it or not) - Dice class - Kids? (bool based having kids or not)	Home - Pay kids - Roll Dice - Maybe get random holiday money on rolls?
Retirement Community - Enough money (bool)	Retirement Community - Game over cue

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

Lawyer office?	
Main game class - Timer for game - Instances of classes above	Main game class - In charge of location??

Class Hierarchy:



Design/Implementation 1: Creating a linked structure with 4 pointers per class, and moving through each structure.

Pseudocode:

Space.hpp file:

```
Class Space {
    Protected:
        String Name
        Space *next
        Space *back
        Space *ptr1
        Space *ptr2
        Friend class List
    Public:
        Constructor()
        Destructor()
        Virtual String getName()
        Virtual bool validMove(Space *)
}
```

Space.cpp file:

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

```
        Constructor
            SET Name = space
            SET 2 ptr Pointers = NULL
            SET next and back = NULL

        Destructor {}

        String getName()
            RETURN name
```

List.hpp file:

```
        Class List {
            Private:
                Space *head
                Int listLength
            Public:
                Constructor
                Destructor
                Void addSpace( Space *, int)
                Void removeSpace(int)
                Void getNodeNames()
        }
```

List.cpp file:

```
        Constructor
            SET head = NULL
            SET listLength = 0
            CREATE Game Layout

        Destructor
            CREATE Space *prev
                SET prev = head
            CREATE Space *ptr
                SET ptr = head

            WHILE (ptr != NULL)
```

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

```
                SET prev = ptr
                SET ptr = ptr's next pointer
                DELETE prev

        Void addSpace (Space *newSpace, int position)
            CREATE Space *ptr
                SET ptr = head
            CREATE Space *prev
                SET prev = head
            CREATE int count
                SET count = 0

            IF (head is NULL)
                SET head = newSpace
                Set head's next pointer = NULL
            ELSE
                WHILE (ptr is not NULL)
                    IF (position equals count)
                        SET prev's next pointer = newSpace
                        SET newSpace's next to ptr
                        INCREMENT listLength
                    SET prev = ptr
                    SET ptr = ptr's next
                    INCREMENT count
            IF (position equals count)
                SET prev's next pointer = newSpace
                SET newSpace's next to ptr
                INCREMENT listLength

        Void removeSpace(int position)
            ……

        Void getNodeNames()
            CREATE Space *temp
                SET temp = head
            WHILE (temp is not NULL)
                DISPLAY temp's getName()
```

```
                    SET temp = temp's next
```

All derived classes in phase 1 will follow the following design:

School.hpp file:

```
Class School : Public Space {
    Public:
        Constructor
        Destructor
        getName()
}
```

School.cpp file:

```
Constructor()
    SET Name = School

Destructor()
    IF school
        DELETE school

String getName()
    RETURN name
```

Design/Implementation 1 Test Plan:

Test Case	Input Value	Driver Function	Expected Outcome	Actual Outcome

Space pointer to School class calls School's getName funct	Space *sp2 = new School;
	Sp2->getName()
		Main()
		getName()
			"School"	"School"

List Class addSpace function adds node to beginning of list	L->addSpace(sp, 0);
	L->getNodeNames();
		Main()
		addSpace()
			"space"	"space"

List Class addSpace function adds	L->addSpace(sp, 0);
	L->getNodeNames();
		Main()
		addSpace()
			"space"
			"School"
				"space"
				"School"

node to second spot of list				

Design/Implementation 2: Going from Singly linked to Doubly linked

List.cpp file was updated with blue highlighted statements as follows:

```
List Constructor()
    SET head = NULL
    SET tail = NULL
    SET listlength = NULL

List addSpace()
    CREATE Space *ptr
        SET ptr = head
    CREATE Space *prev
        SET prev = head
    CREATE int count
        SET count = 0

    IF (head is NULL)
        SET head = newSpace
        SET head’s next pointer = NULL
        SET head’s back pointer = NULL
    ELSE
        WHILE (ptr is not NULL)
            IF (position equals count)
                SET prev’s next pointer = newSpace
                SET newSpace’s next to ptr
                SET newSpace’s back to prev
                INCREMENT listLength
            SET prev = ptr
            SET ptr = ptr’s next
            INCREMENT count
    IF (position equals count)
        SET prev’s next pointer = newSpace
        SET newSpace’s next to ptr
        SET newSpace’s back to prev
```

```
                INCREMENT listLength

        List getNodeNames()
                *NOTE: new while loop only incremented for testing purposes. Will be
                commented or deleted out in finalized code.

                CREATE Space *temp
                        SET temp = head
                CREATE Space *bckwrds
                        SET bckwrds = head
                WHILE (temp is not NULL)
                        DISPLAY temp's getName()
                        SET temp = temp's next

                DISPLAY "traverse backwards"
                SET temp = bcwrds
                WHILE (temp is not NULL)
                        DISPLAY temp's getName()
                        SET bckwrds = temp
                        SET temp = temp's back
```

Design/Implementation 2: Test Plan

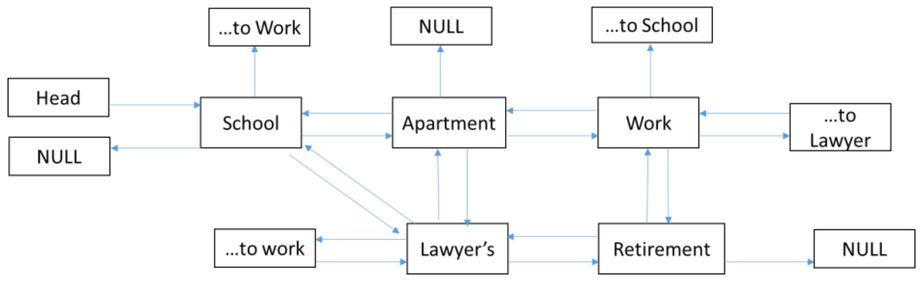
Test Case	Input Value	Driver Function	Expected Outcome	Actual Outcome

Space object back pointers able to traverse backwards to previous node	L->getNodeNames()	Main()
		GetNodeNames()
			"space"
			"School"
			"Apartment"
			"Work"
			"Retirement"
			"Traverse backwards: "
			"Retirement"
			"Work"
			"Apartment"
			"School"
			"space"
				"space"
				"School"
				"Apartment"
				"Work"
				"Retirement"
				"Traverse backwards: "
				"Retirement"
				"Work"
				"Apartment"
				"School"
				"space"

Design/Implementation 3: Interconnecting Remaining Pointers Connections:

Before Apartment is deleted:

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project



Pseudocode:

Added in createGame() function to List class:

List.cpp file:

```
Constructor()
    SET head = NULL
    SET tail = NULL
    SET listlength = NULL
    SET player = NULL

Void List::createGame() {
    CREATE Space *sp
    SET sp to each space type
    CALL addSpace function for each space type
        (This creates a doubly linked list)
    CREATE Space *temp
        SET temp = head

    //Assigning school pntrs to work and lawyer
    SET head's ptr1 = head next, next
    SET head's ptr2 = head next, next, next

    SET temp = temp's next

    //Assigning Apartment to lawyer
    SET temp's ptr1 = head's next, next, next

    SET temp = temp's next

    //Assigning Work to school and retirement
    SET temp's ptr1 = head
```

```
                SET temp’s ptr2 = temp’s next, next

                SET temp = temp’s next

                //Assigning Lawyer to apartment and school
                SET temp’s ptr1 = head’s next
                SET temp’s ptr2 = head

                SET temp = temp’s next

                //Assigning retirement to work
                SET temp’s ptr1 = temp’s back, back
            }
```

Test Plan:

Test Case	Input Value	Driver Function	Expected Outcome	Actual Outcome

Go from school to work node	L->getNodeNames()	Main()
		GetNodeNames()		

Design/Implementation 4: Creating Person Class and Moving Through Structure

Person.hpp file:

```
        Class Person {
            Private:
                String pName
                Double money
                Double studentLoans
                Int kidsNum
                Space *loc
                Bool house
            Public:
                Constructor(string)
                Constructor()
                Destructor()
                String getName()
                Space * getLoc()
                Void setLoc (Space *)
```

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

Person.cpp file:

```
Constructor (string nameIn)
    SET pName = nameIn
    SET money = 100
    SET studenLoans = 0
    SET kidsNum = 0
    SET loc = NULL
    SET house = false

Constructor ()
    SET pName = “Player 2”
    SET money = 100
    SET studenLoans = 0
    SET kidsNum = 0
    SET loc = NULL
    SET house = false

Destructor()

Space * getLoc()
    RETURN loc

Void setLoc (Space *location)
    SET loc = location
```

List.cpp file was updated with blue highlighted statements as follows:

```
Void addSpace (Space *newSpace, int Position)
    CREATE Space *ptr
        SET ptr = head
    CREATE Space *prev
        SET prev = head
    CREATE int count
        SET count = 0

    IF (head is NULL)
        SET head = newSpace
```

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

```
                    SET head's next pointer = NULL
                    SET head's back pointer = NULL
                    CALL player's setLoc()
                         SEND head as parameter
          NOTE: move() and setPlayer() are new functions
          Void move (int input)
               CREATE Space *nwRm
                    SET nwRm = head

               IF (input = 1)
                    SET nwRm = head
               ELSE IF (input = 2)
                    SET nwRm = head's next
               ELSE IF (input = 3)
                    SET nwRm = head's next, next
               ELSE IF (input = 4)
                    SET nwRm = nwRm's next
                    SET nwRm = nwRm's next, next
               ELSE
                    SET nwRm = nwRm's next, next
                    SET nwRm = nwRm's next, next

               CREATE Space *temp
                    SET temp = players location

               CREATE bool vMove
                    SET vMove =  temp's validMove()
                         SEND it to nwRm

               IF (vMove)
                    CALL player's location
                         SEND nwRm

          Void setPlayer (Person *pIn)
               SET player = pIn
```

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project

Test Plan:

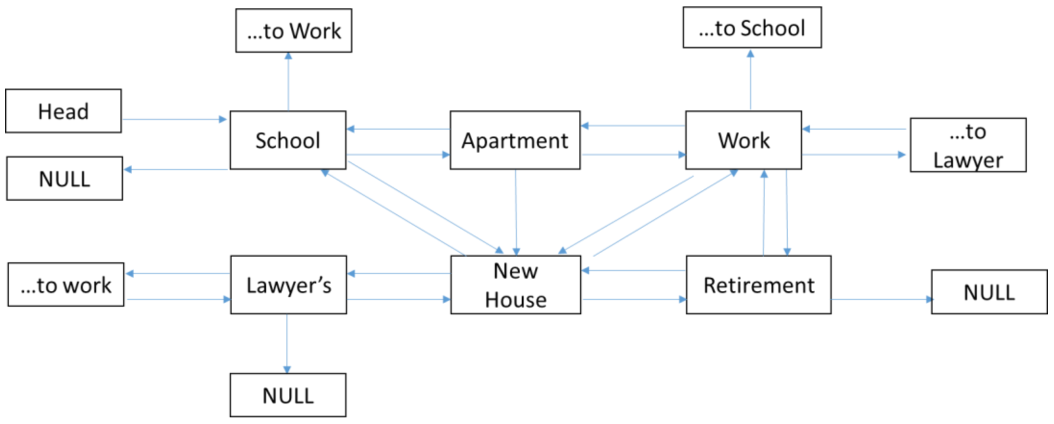
Test Case	Input Value	Driver Function	Expected Outcome	Actual Outcome

Move player's location to same room	1 (School)	Main()
		Move()
		validMove()
		setLoc()
			"Original location: 0x7013c0" (for example)
			"New location: 0x7013c0"
				"Original location: 0x7013c0" (for example)
				"New location: 0x7013c0"

Move player's location to acceptable choice (link to space)	2 (Apartment)	Main()
		Move()
		validMove()
		setLoc()
			"Original location: 0x7013c0" (for example)
			"New location: 0xfc1400" (for example)
				"Original location: 0x7013c0" (for example)
				"New location: 0xfc1400" (for example)

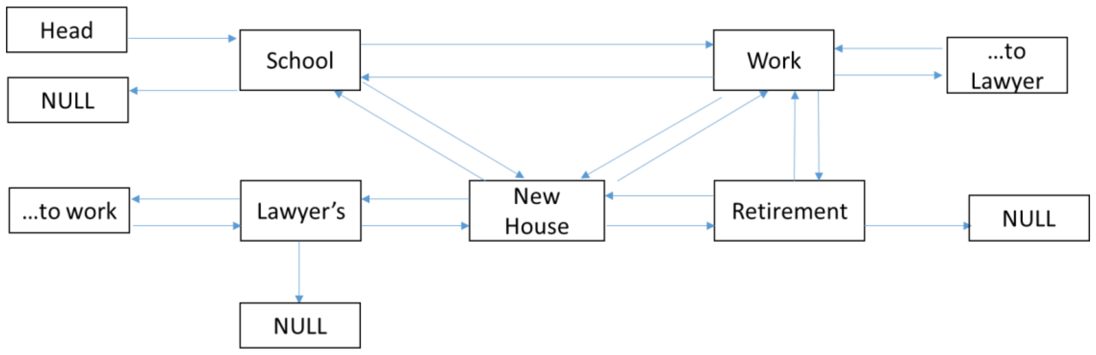
Move player's location to unacceptable choice (no link to space)	5 (Retirement)	Main()
		Move()
		validMove()
		setLoc()
			"That move is not legal. Pick again. "	"That move is not legal. Pick again. "

While house is being added:



After Apartment is deleted:

Name: Molly Arwood
Date: 8-1-16
Class: CS_162_400_Su2016
Final Project


Head
School
NULL
Work
...to Lawyer
...to work
Lawyer's
New House
Retirement
NULL
NULL