Tira Wickland
CS162 OSU
August 7th, 2016

Final Project Design and Testing Plan

I designed my game in multiple steps. I also implemented the program in stages, checking with my design and confirming the functions were operational before continuing. The following document shows my designs and the testing performed on my program. Of note: To make my document more concise I have shown the testing plan after the program was completed to include all my testing properties. As I implemented the program in stages the testing was also performed in stages. I wanted to show the complete testing and so I am showing the testing using the completed version of code.

Design Stage 1:
The first stage of design was designing the game itself. As a text based game, I essentially created a story that a reader could run through like a "Choose Your Own Adventure" story. It was quite fun as I enjoy writing. This document not only came in helpful in following for [1]my design, but also provided me with the text for the majority of the game. Please note that there were some changes made in the text as I found I was wordy at times and did not want the reader to have to do quite so much reading when playing the game. The overall structure of the design remained though. See Appendix A.

Design Stage 2:
Once my text was in place and the story created, I began to design the program itself. My first step in the design was to create the main program, which I decided to call 'game.cpp' to allow my program to be tested and run as I implemented it. I decided it best to implement the game.cpp program and add the introductions and ensure the game.cpp was operational prior to designing the rest of my program.

Design Stage 3: Abstract Base Class
For my game, I created a base abstract class called Cavern. This would serve as my Room class for all derived classes I would build in the future. My attributes of Cavern were as follows: string name, string item, string reqItem, and Cavern pointers north, south, east, and west. This would allow me to link the rooms. For the functions of Cavern, I included a Cavern constructor and virtual destructor (as it is an abstract class). I also had a setName, setItem, setPointers, getName, getText, getItem, getNorth/South/East/West., and action. I used the getText because I wanted an introduction to each room as the user entered it. I knew I would likely need bools to be used as flags/switches throughout the game and help change the behavior of my game, but wanted to design more of my game before determining which bools to add. My pure virtual function was action as each derived class

[1] ** Indicates design change after implementation

would have special actions. Of note, I did not design what I would put in my constructor yet but knew it would take parameters name, item, and requiredItem.

Design Stage 4: Derived Class –Main Room
At this point I designed my derived class Main, to be the main cavern in the game. My main room's action was to have the user call out for his/her roommate (if user decides to). I wanted my attributes to be the same in the derived classes as the base classes at this point. I did not design the special function for this room yet.

Design Stage 5: Derived Class – Mud Room
I designed my derived class mud room to be similar to main room. I did not design the special action function for this room yet.

The base and derived class were easy to design structurally as they were similar to many of previous classes we've created for this class, but the next part required complex design: the linking structure.

Design Stage 5: Linking Structure
For this design, I created several versions of how I wanted to implement the linking class and decided the following design fit my needs the best. I named this class caveSystem.
CaveSystem attributes: Cavern pointers to each room. It will also contain bools to create different behaviors throughout the game. Added a bool addedCave and blockedCave as I knew these would be needed for later when rooms were created and destroyed. I also created a bool isBearThere to help the functionality of the game as the room was added in the future. Other bools would be needed, but I held off at this time. In addition, I created a Cavern pointer current to be the locator throughout the game of the player.
CavesSystem functions: Cave constructor and destructor , findCave, and caveBlocked (these would be most important to design correctly for memory management). I also created a walking(), roomIntro, and doAction() function.
My intent was to have the main actions be managed and performed using this class. This will essentially be my 'controller' class for all the rooms.
At this time I worked on designing the constructor. The order of the design is very crucial as the constructor would have to link the rooms as they were created. Initially I had the setPointers called for each room after the new room was created, but I soon realized my mistake. I would need every room created in order to link them most effectively. I set the order of the constructor to: create room, set pointers, and set the 'current' pointer to main. I also had the bools addedCave and blockedCave set to false and isBearThere set to true.
The destructor of CaveSystem was designed using the bools addedCave and blockedCave. At this point I designed destructor for condition if addedCave and blockedCave are false, delete rooms. Later I can will add other conditions to the destructor for later aspects of the game to manage memory effectively. See Appendix B for sketches of the design, and Appendix C for linked map of rooms.

At this time I implemented the Cavern abstract base class, main room class, and the caveSystem to ensure the logic behind my design was correct. The setPointers function worked as I wanted it with parameters north, south, east, and west begin passed and set to the main room objects.

My walking function had initially been designed to not accept parameters, but looking at the logic after implementation, I realized I would have to have it take the user's choice of direction and changed the function to walking(direction).

Returning to the game file, I then designed the 'travel' through the game.

First I would have the user select a choice of direction, then pass the direction to the caveSystem then move the current pointer. The following is the path the design takes:

Create caveSystem object (with 1 Cavern pointer to Main Room and 1 Cavern pointer to Mud Room)

User menu: Where to next? North, South, East West

User enters character (n, s, e, or w)

Validate character

Call walking(direction)

I ran the program and did not receive any error messages. Now I had to design the caveSystem walking function.

Design Stage 6: Walking function

If character = n

Current = current.->getNorth.

If character = s

Current = current-> getSouth

If character = e

Current = current->getEast

If character = w

Current = current->getWest

I returned to my main design and made a do while loop to continue having the walking function called so the player could travel through the rooms. While my 2 derived classes currently had all pointers set to each other, my design (See Appendix C) has some pointers set to NULL. I would have to ensure if the pointer went to NULL, the player would not move and the 'travel' would be called. So I created a new bool for the walking which I called moved and had my walking function return a bool.

If current = current->getX, moved = true. Else moved = false.

In my main program, designed the order of calling the caveSystem functions.

1. Display Room Introduction (call caveSystem RoomIntro)
2. Do Action in Room (call caveSystem doAction)
3. Travel (call caveSystem walking)

Then using my returned bool, if returned bool is true, do the 3 steps, if bool is false (and player hasn't moved) only display Travel (shouldn't recall room intro and doAction).

As I had setPointer called in the constructor, the logic showed my design worked well and I proceeded to design the remaining rooms.

Design Stage 7: Derived classes Crawl, Treasure, Dark, Bear, and Stalagmites.
I created the derived classes for the remaining rooms (prior to add/blockCave).
I redesigned the caveSystem constructor to create and link all pointers as designed in Appendix C.
RoomIntro function was designed to display name of room getName and room introduction getText. I began to add the text to derived classes' getText and quickly realized the derved classes implementation files would get very 'busy' with the text.

I changed my design to then incorporate a text file to contain most of the text in the game.

Of note, Dark has pointers to main room in all directions except south. This is because I figured in the dark it's easy to get turned around.

Design Stage 7a: Text
I designed the text file to have function calls for various text. The different classes could call the various functions, which would display the text of the class. This made the derived classes much 'cleaner' in appearance. I added a message of "You should not see this" in my abstract base class getText for debugging purposes.

Design Stage 8: Leaving Game
Having navigated the rooms and seeing all logic was in place I wanted to then design the 3 ways the do while loop would end:

1. Game ends if player chooses to leave through 'exit'
2. Game ends if player wins game
3. Game ends if player has made x travels.

I had to design a way for player to leave game on their own accord. I knew this should occur in the main room as this is how my design was set up, but I hadn't yet decided if I wanted a Cavern pointer to leavingGame or a condition in the walking function of caveSystem.

I decided to simplify my program and have a condition where is current = main and direction = n, the condition would run asking user if they wanted to extit. A similar condition would run for later in the game at the end of the lake room where current = lake and direction = s user would be asked if they wanted to exit. This would win the game with or without treasure.

A bool exitGame was also created at that time that would be passed to game through

getExit function I created. The main game could then use the exitGame bool to exit the do while loop.

Design Stage 9: Special Functions Main and Mud
The actions were implemented next.
For the main room:

Special feature is shouting your roommate's name
I didn't want player to keep calling out the roommate's name every time they came in the room, so I had a private member hasDoneAction for the main class and set it to true if the action had been done once.

For the mud room:

Special feature is digging out shovel.
Once shovel had been dug out, the item is collected and put in the bag.
Item is no longer in the room, so the action cannot be performed again.

With these considerations and knowing more rooms would contain items, I added a bool for itemInRoom. This allowed the action function to check for this bool before action performed.

At this time I also had to design the bag the player will have during the game to store the items the player collects.

Design Stage 10: Backpack
I designed a class backpack that contained a static array for the items the user would collect.
The functions addItem, removeItem, displayItem, and checkForReqItem were designed so that parameters could be received and could interact with the other classes through game file.

**Once my design of the backpack was implemented I began trying the logic of the backpack. I soon realized the game I designed did not have many items the user had to collect and the items were used multiple times (such as the shovel). I did not want my player 'dropping' the items because this would make the game unnecessarily challenging (and in some cases impossible) for the player. The removeItem function was removed from my design.

I made the bag accessible through the game file (displayItems) as an additional feature in 'travel' so user could look in the bag before traveling. This would return the bool false so roomIntro and doAction are not performed and the user has the 'Where to next' displayed after exiting displayItems.

Design Stage 11: Special Action Function Stalagmite Room

Special feature: Swinging on the rope

This was by far the hardest special feature to implement because the user couldn't traverse the room without the rope and had to be on the side of the room the rope

was on to use the rope. I had to make another attribute for the Cavern class, side, to be used by the caveSystem class to determine whether the user and rope are on the same side. I put more conditions in the 'walking' function of the caveSystem for the logic to work. I made the special feature only occur if the rope as on the same side and the user could only cross the room by swinging on the rope. If the rope was not on the same side as the user, a message was displayed stating user could not cross and the rope's location was disclosed.

** I changed the rope message to display regardless of whether on the same side as you or not as I found this helped orient player to where they were.
** Player can only swing once on the rope and must leave the room before swinging again.
** I added player side message to display where player was in the room and relation to the rope

Design Stage 12: Map
At this point I wanted a map so the user could see where they were and where they could go. I created a map file designed off my map design Appendix C. This I made accessible to the user via the game file. Similar to viewing the bag, the user could also view the map and actions like roomIntro and doAction would not be performed as the bool would be false.

Design Stage 13: Special Function CrawlSpace
I was able to quickly design and implement Crawlspace using the features in my program I had already created.

Special feature: Pick up flashlight.

This room was designed so the only way back out of the room is through the stalagmite room. Initially I had the Crawlspace space designed so n, e, and w led to other rooms. However, I changed it to NULL after handtracing my logic and realizing that if the player did not pick up the flashlight and took one of those routes they would be unable to return to the Crawlspace because the rope would still be on the north side of the Stalagmite room and the game would be impossible to win. Another challenge I didn't want to make impossible for the user, I instead made n, e, and w pointers NULL so user would have to go back through Stalagmite room and could use the rope again.

Design Stage 14: Special Functions Dark and Bear

Using getReqItem in caveSystem and checkReqItem in backpack, I was able to have the two classes interact in the game file passing the parameters back and forth. If the bag contained the required item, the hasReqItem was set to true in the room. The special features in both rooms would then occur.

In Bear Room, if user has shovel then the user can choose to hit the bear with the shovel or hit the wall with the shovel. Neither of these actions actually change the behavior in the game, essentially a red herring as to how to get rid of the bear.

In Dark Room, if user has flashlight then the user turns it on and the room is illuminated. User can then explore the backpack in the room. Different items in the

bag will produce different texts. User will only be able to interact with one item in the bag and will have to re-enter the room to interact with another.

If user interacts with the Doritos bag, the text displays that the bear leaves. This is a 'game changer' and the bool bearIsThere is now false.

Design Stage 15: findCave

If user is in the bear room and chooses to now travel south, the findCave function is now called.

At this point I created 2 more derived classes, Lake and Treasure. These are similar to the other rooms and I created Lake's special function to have the player converse with the roommate (found at the lake) and if player has the shovel he can decide whether or not they want to continue. If they don't have the shovel they're told to go get it.

I created the findCave to create Cavern pointers lake and treasure and set these with pointers. The lake pointer had all NULL pointers except north, which led to bear. This was intentional as the player must have shovel and decide to get in the boat before continuing with the game. At this point, if the player doesn't have the shovel or doesn't want to take the boat they can still go back through the cave structure.

Design Stage 16: caveBlocked

If user has shovel and gets in boat, caveBlocked now occurs.

I created caveBlocked to delete the cave structure except lake and treasure. The cave is now blocked and the only way the player can go is onward. As mentioned earlier, my destructor had 3 conditions: the first if the original cave structure was in play, the second if the original cave structure and findCave (lake and treasure) were in play, and the third if only lake and treasure were in play.

Once caveBlocked occurred, the lake pointer had setPointer called to change the pointers to the treasure room and remove the bear pointer. This ensured the player couldn't not access a room that's been removed.

** Of note, I had to introduce bools in both the dark room and the lake room that I didn't know I needed when I first designed the program. My design as it originally stood caused seg faults because the findCave and blockedCave could be called multiple times.

Design Stage 17: Counter

A counter was added to game so that if the player travels more than 25 times the game ends.

For a more concise design please see Appendix B.

Design Stage 18: Map
I created a map Appendix C, which I then displayed in C++ to show user where they were in cave structure (not all passageways shown).

Design Stage 19: Menu

Display Title
User enters 1 – Play Game
User enters 2 – Display how to win
User enters 3 – Quit Program

Testing Plan

Test	Room	Action	Expected	Observed
Navigating Rooms	Main	Move north	Display text "you have not found your roommate yet…"	Display text "you have not found your roommate yet…"
		Move south	Bear Room	Bear Room
		Move east	Stalagmite Room	Stalagmite Room
		Move west	Dark Room	Dark Room
	Mud	Move north	Stalagmite Room	Stalagmite Room
		Move south	Bear Room	Bear Room
		Move east	Stalagmite Room	Stalagmite Room
		Move west	Bear Room	Bear Room
	Stalagmites	Move north if on side	Crawlspace	Crawlspace
		If not on side	You can't walk across	You can't walk across
		Move south if on side	Mud Room	Mud Room
		If not on side	You can't walk across	You can't walk across
		Move east if on side	Mud Room	Mud Room
		If not on side	You can't walk across	You can't walk across
		Move west if on side	Main Room	Main Room

		If not on side	You can't walk across	You can't walk across
	Crawlspace	Move north	Can't go	Can't go
		Move south	Stalagmite Room	Stalagmite Room
		Move east	Can't go	Can't go
		Move west	Can't go	Can't go
	Dark	Move north	Main Room	Main Room
		Move south	Bear Room	Bear Room
		Move east	Main Room	Main Room
		Move west	Main Room	Main Room
	Bear	Move north	Main Room	Main Room
		Move south	Blocked if bear there	Blocked if bear there
			Lake if bear gone	Lake if bear gone
		Move east	Mud Room	Mud Room
		Move west	Dark Room	Dark Room
	Lake	Move north before blocked	Bear Room	Bear Room
		After blocked	Blocked	Blocked
		Move south before blocked	Can't go	Can't go
		After blocked	Display do you want to leave cave?	Display do you want to leave cave?
		Move east	Can't go	Can't go
		After blocked	Can't go	Can't go
		Move west	Can't go	Can't go
		After blocked	Treasure Room	Treasure Room
	Treasure	Move north	Can't go	Can't go
		Move south	Can't go	Can't go
		Move east	Lake Room	Lake Room
		Move west	Can't go	Can't go
Navigation	User enters 'n' User enters 's' User enters 'e' User enters 'w'		getNorth() getSouth() getEast() getWest()	getNorth() getSouth() getEast() getWest()
Special Actions	Main	User enters 1	Shouts	Shouts

		User enters 2	Remains silent	Remains silent
	Mud	User enters 1	Digs, Shovel added to bag	Digs, Shovel added to bag
		User enters 2	Shovel stays in room	Shovel stays in room
	Stalagmites	If rope is on same side	User can cross room, rope and user on that side of room.	User can cross room, rope and user on that side of room.
		If rope not in same side	Can't cross room. Message showing where rope is is displayed	Can't cross room. Message showing where rope is is displayed
	Crawlspace	User enters 1	Picks up flashlight	Picks up flashlight
		User enters 2	Flashlight remains in room	Flashlight remains in room
	Dark	If user has flashlight	Display backpack and user can use items in backpack	Display backpack and user can use items in backpack
	Bear	If user has shovel and enters 1	Display message about not wanting to hit bear	Display message about not wanting to hit bear
		If user has shovel and enters 2	Display message about hitting wall	Display message about hitting wall
	Lake	If user has shovel	User may get in boat (presses 1) or remain on shore (presses 2)	User may get in boat (presses 1) or remain on shore (presses 2)
		If user does not have shovel	User is prompted to get shovel	User is prompted to get shovel
	Treasure	User enters 1	Picks up treasure	Picks up treasure

		User enters 2	Treasure stays in room	Treasure stays in room
User validation	If user enters character other than those specified		Display error message and ask for character from user	Display error message and ask for character from user
findCave()	Dark Room	If user uses eats Doritos	Bear leaves, Lake and Treasure Rooms created	Bear leaves, Lake and Treasure Rooms created
caveBlocked()	Lake Room	If user has shovel and enters boat	Main, Mud, Stalagmites, Crawlspace, Bear, and Dark Rooms deleted	Main, Mud, Stalagmites, Crawlspace, Bear, and Dark Rooms deleted
Times Up	Counter = 15		Display message about time running out	Display message about time running out
	Counter = 30		Display message that time ran out, end game	Display message that time ran out, end game
Exit Game	If user chooses to exit game by going north from Main room		Exits game	Exits game
	If user chooses to exit game by going south from Lake room		Display end sequence Exit game	Display end sequence Exit game
User Menu	User enters 1		Plays game	Plays game
	User enters 2		Display how to win	Display how to win
	User enters 3		Quits game	Quits game
Bag	User enters ‘b’		Bag is displayed with items	Bag is displayed with items
Map	User enters ‘m’		Map is displayed	Map is displayed

How Did You Understand The Problem?
I created the story before I began the design of the program. Once the story was created I was able to understand the problem and design it accordingly.

How Did You Approach The Design?
After the story was created I designed the game (in stages) and implemented the stages before going on to the next stage of design. By implementing each stage before moving on with the next stage of design, I could check my logic and ensure that each part of the program is working as it should.

What Was The Hardest Part For You?
I had difficulty with the game changing behavior throughout the game. My original design did not have as many bools as needed and I had to add more bools to make the program behave differently at different times. Logic testing (most of the time handtracing but sometimes testing with implementation and cout statements) helped me see where the lacking bools were still needed.

Appendix A:

Your roommate didn't come home last night. You search his room and find topographic maps and sketches of cave structures. History books are piled up on his desk with pages of newspaper articles from 1890s taped to his walls. One article catches your attention: MAN'S FORTUNE VANISHED IN COLLAPSE. It discusses a man who, in his greed, hid his fortune underground in a cave, only to have the cave collapse later that year. The article mentions a reward for the treasure. Other newspaper clips sport the reward ad several more times. These clippings date 1900-1901. The last taped news clipping is an article from 1930, which discusses the 'legend of the buried treasure'—the man's fortune never claimed. Oh boy…your roommate's at it again and he's known to get himself into trouble. You examine the topographic map closer and sure enough, he's drawn an X miles out of town with one word confirming your fear: BINGO. You grab the topographic map, cave sketches, and your keys. Time to be a hero.

You arrive at the 'Bingo'-marked X by midday. Sure enough, you see your roommate's car parked near a pile of boulders. A tree next to the vehicle has the end of a rope tied around it. The other end of the rope leads into the boulders. You park your truck beside his car and check your cell phone. No signal. Were you expecting anything different? You hop out of the car and approach the pile of boulders. Nimbly climbing up the closest boulder, you peer over the edge to see the rope leads right over the edge of the boulders into a gaping hole. The black hole seems to go on forever, with no light escaping. How far down does it go? How far does the rope go? You call down to your roommate. No response. You return to the tree and check the knot. Seems solid.

1. So now you face a tough decision:

Do you descend into the unknown to help your friend? -> onward

or

Drive back for help? -> You decide to go for help. And why not? You're a responsible fellow. You go back to your car and hear the cringeworthy hiss of a flat tire. Sure enough the tire is completely flat. Just your luck. You check the door of your roommate's car to find it locked. You peer in the windows to search the car for his keys, just in case. Nope. Ok so you're down to one decision…descending into the unknown.

Sitting on the edge of the boulder gripping the rope tightly, you say a quick prayer…well really it's just a lot of swear words with your roommate's name amongst them, but whose listening? You take a deep breath and begin sliding down the rope. Your back scrapes along the rock as you slide down the boulder and then

the rock against your back is gone and you're suspended in air from the rope. You hang on now for dear life, waiting for your eyes to adjust, scared to slide any lower. What has your roommate gotten you into? A moment later your eyes adjust and you find yourself in a large cavern. The ground is about 20 feet below you. You continue sliding down the rope, pleased to see it does extend to the ground. When your foot hits solid ground you can't help but smile. It was almost…fun. Your eyes are now well adjusted to the cavern and you realize you can actually see quite well. Sun light streams into the room and you see the walls of the cavern look like wet bubbling mud. Feeling the closest you find it smooth, dry, and cool. The air seems cooler after touching the stone. You decide you do not want to be in this cave after dark and return to your mission of finding your roommate.

Do you shout to your roommate?
1. Yes -> bats enter room
2. No -> remain silent

You walk along the walls of the cavern and notice there are narrow openings along the wall.

MAIN CAVERN
Do you walk
North -> ASCEND: You haven't found your friend yet. Are you sure you want to leave the cave? Ascending will exit program.

West -> DARK: The passageway gets darker and darker until you are only able to guide yourself with your hands touching the walls on either side of you. Then the walls drop away as you enter a room. The room is pitch black. You take a step into the room, but immediately you feel gut-clenching fear. How will you find your way out. You scramble backwards into the safety of the passageway and hurry back to the main cavern.

East-> Stalagmites You duck to avoid the ceiling as the passage gets smaller and smaller. Then it opens to a large, open space. You're standing on a ledge about 10 feet up from the floor of the cavern. You stare down at the hundreds of stalagmite jutting up from the ground of the cavern. That looks challenging to cross. You notice a rope dangling from the center of the room. Cursing under your breath, you begin to wonder how you ever got Indiana Jones as your roommate. The end of the rope is caught on a rock beside a ledge on the east side of the room. There's other ledges on the north and south sides of the room. All three ledges seem to have passages leading out of the room to parts unknown. Time to start exploring. But first you have to get to the rope and you won't be able to reach it from this ledge. Better head back to the main cavern and find another way.

South -> BEAR CAVERN: You walk down the passageway to find another large room. You barely notice much of the room though as your attention is focused on the big furry butt of what can only be a bear lying on the far side of the room. You walk

cautiously into the room, trying to see around the behind of the beast. He looks like he's stuck with his front half in another passageway.

If 2a. You clutch the shovel a little tighter. What if the bear ate your roommate? What if your roommate is trapped behind the bear? You whisper your roommate's name. No response. You need to see what the bear's blocking. Do you

2a1 Hit the bear's rear with the shovel? -> Are you crazy! You quickly disregard this idea, not wanting to end up bear bait. There has to be another way…

2a2 Bang the shovel against the cave wall? -> You bang the shovel lightly against the wall and when that doesn't get a response you hit it a little harder. The bear grunts, but doesn't budge.

If 2b What if the bear ate your roommate? What if your roommate is trapped behind the bear? You whisper your roommate's name. No response. You need to see what the bear's blocking. Maybe there's an item in another room that can help you.

There are passageways to the NORTH, BACK, WEST, and EAST. Where do you go?

North –Main Cavern

West – DARK (flashlight==no): The passageway gets darker and darker until you are only able to guide yourself with your hands touching the walls on either side of you. Then the walls drop away as you enter a room. The room is pitch black. You take a step into the room, but immediately you feel gut-clenching fear. How will you find your way out. You scramble backwards into the safety of the passageway and hurry back to the bear cavern.

Back -> MAIN CAVERN: You head back to the main cavern.

East -> MUD: You start to feel your feet squish and you feel yourself going deeper into the earth as you walk down the passageway. As the passageway opens up to a room, you find yourself in ankle-deep mud. You begin to cross the room and trip over something in the room. It's covered in mud.
Do you dig the item out?

Y –You grab hold of the object and pull. Slowly it emerges from the mud. A shovel! This could come in useful.

N- Not wanting to get your hands dirty, you decide to move on.

Do You go North: Stalagmites, South: BEAR CAVERN, West: Bear Cavern East: Stalagmites

Stalagmites

If (rope == on your side)

Do you swing…

1. Yes – which direction: north, south, west?

South-rope now on south side (rope south). Go through passage -> MUD

West-rope now on west side (rope west) go through passage->MAIN CAVERN

North-rope now on north side (rope north). Go through passage-> CRAWLSPACE (flashlight==no) As you crawling through the passage your hand brushes against a cylindrical, cold shape. You grasp it and realize its your flashlight that's been missing. No wonder you couldn't find it! Having found another reason to yell at your roommate when you find him, you click on the light and continue along the passageway. (flashlight==yes) You put it in your pocket for safekeeping and grab the rope.

2. Leave rope -> return back to room you were in

DARK ROOM (flashlight==yes) Clicking the flashlight on, you shine it across the room. A neon orange backpack that can only be your roommate's is lying on the ground in the cave. Your roommate is nowhere to be seen. You search through the backpack. A bottle of water, a cell phone, and a bag of Doritos.

3. Do you:

Leave bag? -> You decide to leave the bag. Your looking for a roommate, not a bag. The flashlight begins to flicker. Time to go.

drink water? -> Refreshing =)

Use cellphone? -> No signal. By now you should really know better

Eat Doritos? -> The moment you tear open the bag you hear a growl from the other room. Seriously? A bear comes barreling into the room and swipes the Doritos bag from you. Not feeling up to wrestling a bear for his Doritos, you watch him lumber away with his prize. (bear = gone)

Do you go:
MAIN CAVERN

BEAR CAVERN- You can see a passageway to the south of the room and follow it. It leads to a huge cavern, complete with an underground lake! And there, sitting on the shore of the lake, is a familiar face grinning up at you.
"Boy am I glad to see you!" Your roommate calls out.
He stands up and steps into a boat. Wait a boat? He brought a boat down here? How long has he been planning this?
If shovel,-> "You brought the shovel right? I Are you ready?" He asks you.

1. Yes 2. No.

"Did you find the shovel? No? Dude we need the shovel. Go get it!"

1. If Yes = Perfect, he exclaims as you hop in. Do you want to paddle or should I? You meet his question with a glare and he chuckles. Ok, I'll paddle if you navigate.

You begin to cross the lake. There are tunnels leading to the west, east, and to the south you see the shoreline.

West- you paddle west into the tunnel on the west side. Why not explore a little in this boat. The tunnel gets dark and you click on your flashlight. Shiny reflective surfaces hit your eyes, causing you to blink them shut. Squinting, you open them. On a recess of the wall is boxes of gold coins…gems…jewels. So much glittering! You can't help but grin. The treasure! Your roommate lets out a whoop and paddles over to the walle.
Do you take the treasure?

Yes (treasure == yes)

No (treasure == no)

What a fortune! You load the boat with the treasure. It's heavy, but the boat stays afloat. You run your fingers through the boxes of coins and jewels. Your roommate suddenly doesn't seem as crazy to you. He has began paddling again and you soon emerge out of the tunnel to find yourself back in the lake. This time with a big smile on your face. Where to? He asks.

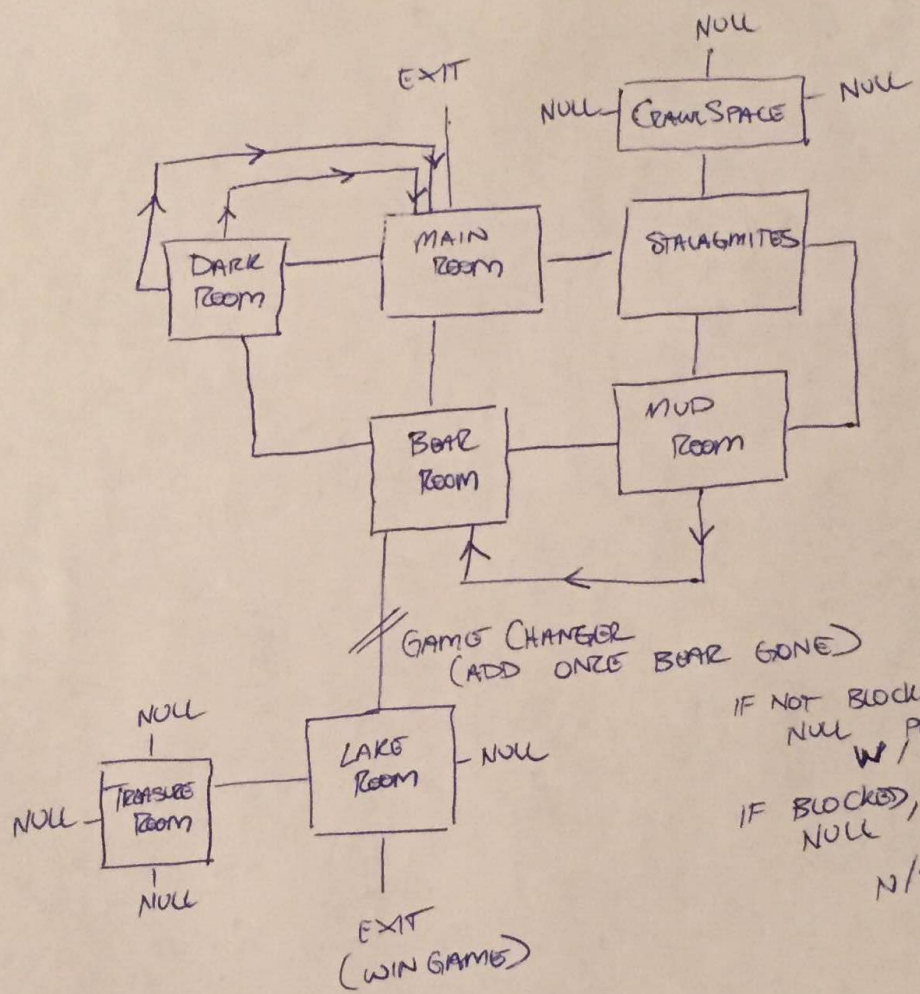
South – you cross the lake in a hurry. The boat has barely touched the shore before you jump out, eagerly eyeing the slanted slope of shore leading upwards. (if treasure == yes)-you strip off your shirt and use it to wrap up the boxes of treasure. No way you're not leaving it behind. Your roommate follows your lead. Then the two of you begin up the shore, holding your shirts carefully so no treasure spills out.

Sure enough you see a passage at the far end with faint light shining in. You're getting out of here! You and your roommate charge up the slope (well you charge and your roommate hobbles quickly after you). You journey up the passageway for what feels like hours but is probably only a minute or two. Then…sky! You pop out of a hole in the ground and throw yourself onto the forest floor of pine needles. You've never been so glad to see pine trees. Your friend heaves himself out after you, lying in the pine needles and staring up at the sky.

If (treasure==yes) You look at the treasure lying beside you in your shirts. Not bad for a day's work. Your friend laughs. Boy was that an adventure, he says, panting. So what are you doing next weekend?

If (treasure == no) Boy was that an adventure. He says, panting. So what are you doing next weekend?

Appendix B


NULL
EXIT
NULL
CRAWLSPACE
NULL
DARK ROOM
MAIN ROOM
STALAGMITES
BEAR ROOM
MUD ROOM
GAME CHANGER
(ADD ONCE BEAR GONE)
NULL
NULL
TREASURE ROOM
NULL
LAKE ROOM
NULL
EXIT
(WIN GAME)
IF NOT BLOCKED, LAKE HAS NULL POINTERS IN W/E/S DIRECTIONS
IF BLOCKED, LAKE HAS NULL POINTERS IN N/S/W DIRECTIONS

Appendix C

Concise Designs of Files/Functions

*Note simplistic functions such as accessor functions not discussed

Game File Design:

Display menu options for user
    1. Play game
    2. How to win
    3. Quit program
Display Intro
Get user decision on whether to enter cave
Enter cave
Create caveSystem
Create backpack
Do
    Check if player has left room since last turn
    Room Introduction
    Check if item room needs is in backpack
    Set bool found in room if item in backpack

    Do action in room
    Display user with options to navigate, display map, or display bag
    Get user direction
        If m -> display map
        If b -> display bag
        If n/s/e/w -> determine/travel that direction
    Set bool of whether player left room
    Increase counter
Continue do while loop until counter = 25 or exitGame bool is true

If counter = 25
    Display time up message
Display menu options for user
Continue until user enters 3

CaveSystem File Functions

CaveSystem Constructor
- Create MainRoom Pointer
- Create MudRoom Pointer
- Create Stalagmites Pointer
- Create BearRoom Pointer
- Create DarkRoom Pointer
- Create CrawlSpace Pointer

Set lake pointer of CaveSystem class to NULL
Set treasure pointer of CaveSystem class to NULL
Set bools addedCave, blockedCave, and hasTreasure to false
Set pointers of each room (N/S/E/W pointers)
Set Current Pointer to Main
Set exitGame to false and isBearThere to true

CaveSystem Destructor
- If addedCave is false and blockedCave is false
  - Delete main, mud, stalagmite, bear, dark, and crawl
- If addedCave is true and blockedCave is false
  - Delete main, mud, stalagmites, bear, dark, crawl, lake, and treasure
- If addedCave is true and blockedCave is true
  - Delete lake and treasure

Walking
- Check whether player is in main room and going north
  - If true, see if player wants to leave game (y or n)
    - If y set exitGame bool to leave game
- Check whether player is in lake room and going south
  - If true, see if player wants to leave game (y or n)
    - If y set exitGame bool to leave game
- If not exiting game
  - If going east and east is not NULL
    - Check if trying to cross Stalgmite room
      - If trying, display error message
    - Else move east
    - If now in stalagmite room, set side of room
  - Else If going west and west is not NULL
    - Check if trying to cross Stalagmite room
      - If trying, display error message
    - Else move west
  - Else If going north and north is not NULL
    - Check if trying to cross Stalagmite room
      - If trying, display error message
    - Else move north
    - If now in stalagmite room, set side of room
  - Else If going south and south is not NULL

Check if trying to cross Stalagmite room
If trying, display error message
Else move south
If now in stalagmite room, set side of room
Else If in bear room traveling south and south is NULL
Display bear blocking message
Else If in lake room and cave not blocked and trying to travel s/w/e
Display need boat message
Else if in lake room and cave blocked and trying to go n
Display bear blocking message
Else
Display space too small
Set moved bool determing whether the player moved during turn

RoomIntro
Display Room Name
Display Room Text

DoAction
Perform room action
If bear is gone after action
Set bool that bear is gone
Call findCave function to add lake/treasure rooms
If blockCave after action
Call caveBlocked function to delete rooms
If treasure after action
Set bool that player got treasure

FindCave
Create lake and treasure Rooms
Set lake and treasure room pointers
Set addedCave bool to true (for CaveSystem Destructor)

CaveBlocked
Delete main, mud, stalagmite, bear, dark, and crawl.
Set blockedCave bool to true
Set pointers for lake room (modify the pointers to ensure no seg faults)

CrawlSpace Action Function
If Flashlight in Room
Display Flashlight text with player choice to take flashlight
If player takes flashlight, add flashlight to bag (pass string)
Remove flashlight from room

Lake Action Function

If Player has Shovel and Cave not Blocked yet ask if player wants to set sail
If yes, block cave

Treasure Action Function

If Treasure in Room
Ask player if they want to take treasure
If yes, add treasure to bag, remove treasure from room

Dark Action Function

If player has flashlight and hasn't eaten the Doritos (have to check bool to avoid seg fault)
Ask player if they want to drink water, use cellphone, or eat Doritos
If player opens Doritos, bear leaves
Set bear is gone bool (used in DoAction function)

Bear Action Function

If player has shovel and the bear is there
Ask user if they want to hit bear or wall with shovel
Display messages

Stalagmites Action Function

If player on same side as rope
Ask player if they want to swing
Move player across room
Display which side of room player's on
Display which side of the room rope's on

Mud Action Function

If shovel in room
Ask player if they want to dig it out
If yes, put shovel in bag and remove shovel from room

GetText Function (in most rooms)

See if firstTime bool is set
Display firstTime text if true
Display other text if false

Backpack File

Item strings stored in array (max size = 3)
Display items iterates through array
AddItem adds item string to end of array
CheckForReqItem(string) iterates through array searching if string == item in array. If string is in array, return found is true