Steps 4 and 5

Step 4 completes the project. Step 5 is the final turn-in, and allows you to correct any errors in Step 4. If you are satisfied with your Step 4 grade *you should not turn in Step 5*, and your Step 4 grade will be used for your Step 5 grade. If you turn in your Step 4 early you will 1.2 points of extra credit on the project for each day it is turned in early, for up to 6 points total added to your project.

You should make sure you have the latest version of the .xml files before beginning Step 4 – I've changed a few values in them to make things work better.

In Step 3 we did player movement, attacks ignoring weapons and armor, picking up and dropping items, and the inventory command. For Step 4 we will do the following:

Player Movement:

Player movement serves two purposes in the game. Movement obviously has the effect of either moving the player or causing the player to hit a monster. Movement also serves as a measure of time. *Every so many moves the player gains one hit point added to its health.* The number of moves that does this is the hpMoves element in the .xml file description of the player. The Hallucinate action also lasts for a given number of moves that is given by the actionIntValue element in the .xml file. So it is necessary for movement to be communicated to both the object that controls player movement (the Player object, in my game) and to the Hallucinate action. **Enhancements to combat:**

When a victim creature, either the player or a monster, is hit, all hitActions associated with the victim creature are performed by that creature.

When a creature, either the player or a monster, dies, all deathActions associated with the dying creature are performed by that creature. When the player dies the game is over. The screen should be displayed as it is after the player's deathActions are performed.

Computing damage from hits and damage done when hitting:

If the player wields a sword, the value of that sword is added to the hitpoints computed in Step 3. If the player is hit, the number of hit points inflicted on the player should have the value of worn armor subtracted from it. Note that armor or a sword in the pack that is not worn or wielded as no effect.

Implement all commands:

Change, or take off armor 'c': armor that is being worn is taken off and placed it in the pack. If no armor is being worn a message should be shown in the *info* area of the game display.

Drop 'd' <integer>: drop item <integer> from the pack, where <integer>-1 is an index into the pack container. If the index does not refer to an item in the pack an informational message is printed on the game display in the *info* area. If the item is dropped it should no longer be shown as in the pack until it is picked up again. If the item dropped is a wielded sword or worn armor, the sword or armor is no longer wielded or worn.

End game 'E' <Y | y>: end the game. Print a message in the info area using Char objects added to the objectGrid to make it clear why the game ended.

Help: '?': show the different commands in the *info* section of the display. This is the bottom message display area.

Help 'H' <command>: give more detailed information about the specified command in the *info* section of the display.

Show or display the inventory 'i': show the contents of the pack, printing the *name* for each item in the pack. For swords and armor print out the value of the sword or armor. For scrolls print out what the scroll name. Each item is preceded by an integer 1 … *max items in the pack* that is used to index into the pack container when removing or dropping items in the pack. If an item in the inventory is a sword that is wielded, put "(w)" after the name of the sword. If the item in the inventory is armor that is worn, put "(a)" after the name of the armor.

Pick up an item from the dungeon floor 'p': pick up the item on the dungeon floor location that the player is standing on and add it to the pack container. Room and passage way floors cannot be picked up. If multiple items are in the location, only the top item is picked up.

Read an item 'r' <integer>: the item specified by the <integer>-1 must be a scroll that is in the pack. Reading a scroll causes the ItemActions associated with the scroll to be performed. The scroll is removed from the pack when it is read, and basically vanishes from the game.

Take out a weapon 'T' <integer>: take the sword identified by <integer>-1 in the pack container and have the player wield it. If the identified item is not a sword, or no such item exists, show a message in the *info* area of the game display. The sword remains in the pack when wielded – this makes it easy to drop sword that is wielded. If we want to no longer wield the sword it needs to be dropped and picked up.

Wear item 'w' <integer>: take the armor identified by <integer> from the pack container and make it the armor being worn. If the identified item is not armor, or no such item exists, show a message in the *info* area of the game display. Armor is taken off with the 'c' command, described above.

Implement Creature Actions:

All actions, both death and item actions, should print something showing that they executed.

Creatures can have the following actions associated with them.

Print: *this is mentioned in the RogueProjectInstructions.pdf file but we will not be implementing it.*

ChangeDisplayType: changes the character that represents the creature. The new character is specified by the actionCharValue element in the ChangeDisplayType XML description. Used primarily in a death action for the Player to change what a dead player looks like.

Remove: removes the creature from the display. Primarily used to remove a dead monster from the objectGrid and the terminal.

Teleport: causes the creature to go to a random place in the dungeon that is legal for a creature to be, i.e., within a room or a passageway between rooms. A creature must end up in a part of the dungeon that it is legal for a player to move to.

UpdateDisplay: causes the creature to update the display of itself. If the creature is the player, the hitpoints displayed in the top message area should also be refreshed.

YouWin: This is typically executed by the creature that is killed. We will print out the message given in the actionMessage element of the XML definition of the action.

Players have the following additional actions:

DropPack: drop the item in position 0 of the pack container. The action of dropping is the same as the drop command above, under Commands. If the pack is empty this does nothing.

EmptyPack: the same as DropPack, except items are dropped repeatedly until the pack is empty.

EndGame: typically executed when the player dies, it causes the game to be ended and all further input to be ignored. I do this by setting a static flag in the Dungeon that signals the end of the game.

Implement Item Actions:

Bless/curse item: This action is associated with scroll. It reduces the effectiveness of a sword or armor that is being worn or wielded. A sword or weapon in the pack is not affected. The actionCharValue in the XML definition of the action will be 'a' if it affects the armor being worn, and a 'w' if it affects the sword, or weapon being worn. If the item to be affected is not being worn or wielded, the scroll has no effect. Print a message to the bottom message area indicating that the scroll was blessed or cursed. I print the message:

scroll of cursing does nothing because " + name + " not being used when the object is not worn or wielded, and *name + " cursed! " + intValue + " taken from its effectiveness*, where *name* is the name of the item.

Hallucinate: it causes each item within the dungeon to display a different character with each move of the player. Empty spaces outside of Rooms and Passages display normally. Displayed character should be among the set of characters used to represent room floors, room walls, passage ways, passage junctions, items and creatures. This lasts for a limited number of moves specified in the actionIntValue element for the action in the .xml file at which time objects in the terminal are again displayed normally.

When it is invoked it will print out a message in the info area of the game display that hallucinations will continue for some number of moves.

What to turn in

Turn in your code in a directory named after your userid.

If you are in the C++ class, have a make file under the userid to run your code. If you hardcoded your file name, then

1. learn how to use command line argument in C++, because not doing it is an amateur move and not very hard;
2. put in a README.txt file saying how to change the filename.

If you are in Java class there should be a src directory immediately under the userid directory. From the userid directory, javac src/Rogue.java should build your program, and java src.Rogue <xml filename> should run your program with the filename.

For both Java and C++, inside the userid directory there should be a video of your running the program. The video should contain the following, in this order.

A run of your game with badScroll.xml. You should run the program,

1. do an inventory that shows the sword;
2. read the scroll, and do an inventory that shows the sword. Note that your inventory should show the strength of the sword;
3. Wield the the sword and do an inventory.
4. Drop the sword, pick it up and show the inventory.

A run of your game with death.xml. Fight the troll and let it kill the player, showing that the player ChangeDisplayType, UpdateDisplay and EndGame death actions are performed.

A run of your game with dropPack.xml.

1. Pick up the two armors and the sword and put them in your pack.
2. Put on and take off the +10 armor.
3. Try to wield the armor.
4. Try to wear the sword.
5. Drop the three items in three different locations in the dungeon and do an inventory.
6. Pick up the three items and do an inventory.
7. Drop an item outside the range of pack items.
8. Drop the three items in one location and do an inventory.
9. Move off the location where the items were dropped and then and back onto the location.
10. Pick up all three items up and do an inventory.
11. Fight the troll and do an inventory to show that an item is dropped from the player's pack when it is hit.

A run of your game on dungeon.xml. Fight and kill a monster.

A run of your game on hallucinate.xml.

1. Pick up and read the hallucinate scroll.
2. Move 5 times showing the change in displayed characters.
3. Move a sixth time and show that the hallucinations have stopped.