# *Adventure*

**Premise**: *Adventure* was a computer program that introduced a whole new kind of game to the world in the late 1970s. The player navigates a complex castle, picking up tools and treasures while avoiding various obstacles (tests, homework programming assignments, etc.) and warding off various evil beings (like politicians and computer science professors). From any room there may be passages to another room in any of the directions: east, west, north, or south. A passage from one room to another doesn't guarantee a passage in the opposite direction, and even if there is a passage in the opposite direction it doesn't necessarily open in the opposite direction – a passage that leaves east from one room may hook and enter at the north side of another room. Associated with each room is a text description of its name. A castle of rooms is easily represented as a directed graph. You will use a linked graph to implement a program that constructs a moderate size adventure castle and allow a user to navigate around it.

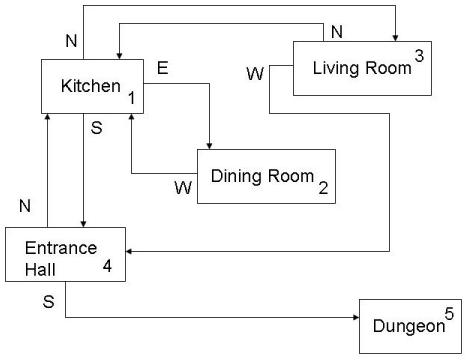
***Your goal is to find the King's crown and retrieve it for him.***

1. Rooms will have doors, zero or one characters and zero or more items (see below).
2. A passage from one room to another is represented as an arc between graph nodes containing the rooms. The label of the arc indicates the direction in which the player leaves the room to traverse the path represented by the arc.
3. Note that the graph implementation assumes arcs are equally weighted.
4. The user starts in the Entrance Hallway. Then use a loop that allows the user to interactively explore the castle by entering one-letter directions (ex., E,e,W,w,N,n,S,s,F,f,D,d,Q,q). Tell the user what room he/she is in or that there is no door in the selected direction, or perform the action requested by the input (see below).
5. Tell the user what and/or who is in the room with you.
6. No one is perfect, and after running 17 programs, it is possible that one can press the wrong key. Handle this elegantly. Case is also not an issue. 'N' and 'n' should be treated the same.
7. Rooms will be numbered consecutively, starting at 1, with a maximum of 98 rooms.

*Here is an example for the input file: castle.dat*

- node and description: an integer followed by a space and up to 20 characters for the room name
- 99 indicating no more rooms
  - followed by triplets (room#, room#, direction): integer, integer, 1 character
- 99 indicating no more door information
  - followed by room #, number of items in room, list of items in room

1 Kitchen
2 Dining Room
3 Living Room
4 Entrance Hall
5 Dungeon
99
1 3 N
1 2 E
1 4 S
3 1 N
2 1 W
4 1 N
3 4 W
4 5 S
99
1 1 nothing
2 3 crown wizard jewel
3 4 vampire candle club crown
4 2 table elixir
5 2 werewolf tome
<EOF>




*This is a sample only useful for clarifying file input format. You will want to design something more interesting to see if the game works.*

---

**Commamd List:**

| | |
|---|---|
| N,n | North – move through the north door if there is one |
| S,s | South – move through the south door if there is one |
| E,e | East – move through the east door if there is one |
| W,w | West – move through the west door if there is one |
| C,c | Contents – list the contents of the knapsack |
| U,u | Use – use an item from the knapsack |
| T,t | Take – pick up the item (same as Use) |
| R,r | Remove – remove an item from the knapsack (same as Use) but leave in room |
| F,f | Flee – take the shortest path to the entry hallway to exit the castle |
| Q,q | Quit – you are tired of playing the game |
| D,d | Debug – this is so that I can verify you built your graph correctly<br>output should be a list of rooms and connections, one per line<br>example: *kitchen north to living room* |

---

**Character List:**

*Wizard* – This is a good guy. He will give you a spell. You do not need to defend yourself against him. He will also give you 5 points for health.

*Hag* – She can be good or bad depending on your perspective. You must give her something out of your knapsack or she will take 10 points from your health. You should present the contents of the knapsack and let the user decide what to give her. Spells do not work on her.

*Troll* – This is definitely a bad guy. You can get past him unharmed with a club or a spell. Otherwise he takes 10 points from your health and bruises you pretty bad, too.

*Vampire* – A no-brainer bad guy. If you have garlic, you can pass without losing health. If you have a wooden stake, you can kill him. If you kill him, he is removed from the room. If you have no defense, your blood is sucked at the rate of 25 health points. Spells don't work on vampires.

*Werewolf* – Again, not a good thing to run into. His bite is lethal. You die. If you have a silver bullet, you can kill him (and remove him from the room). You can slip by him with a spell.

*Hunchback* – A harmless sort of bloke. Doesn't hurt you, but he steals something from your bag. What gets stolen is random. You do not get to chose. They're just too dumb for spells to work on.

---

**Things you may come across in the rooms:**

*Defensive items:*

garlic
spell (sometimes they are found on a table – the wizard has an endless supply)
club
silver-bullet
wooden-stake

*Health items:*

elixir (good for 25 points of health, only when you apply it)
bread (good for 10 points of health, only when you apply it)

*Fairly useless items except for trading with the hag or being stolen by the hunchback*

jewel
tome
goblet

*Things you may find but you cannot move*

nothing
table
candle

*Your goal:*

crown

---

**Health regulations:**

You start with 100 health points.
You can never exceed 100 health points, so if you use elixir that would make it more than 100, the elixir is wasted.
When you move to a new room, your health should be displayed.
When something happens to change your health, it should be displayed.
When you drop to zero (or below) you die.
If you are bitten by the werewolf, you die.

---

**Rules of engagement:**

- Once a spell is used, it is spent. It simply goes away.
- You can keep and reuse garlic and the club.
- The silver bullet and wooden stake may only be used once.
- You can only carry three things in your knapsack. This includes everything (spells, elixir, etc.)
- There may be more than one of each character except the werewolf and vampire. There are only one of each.
- You may flee to the entry hallway. You are running so fast, that the characters in each of the rooms along your path cannot hurt you.
- However, once at the entry hallway, you may not reenter the castle. If you have the king's crown, you win the game. If you do not, your head is chopped off by the King's executioner who is waiting for you at the door. There's a lot of blood and gore, and all the regular stuff.

---

**Rules of the program:**

- You must use a graph data structure.
- You may choose either an adjacency list or an adjacency matrix. (I recommend the matrix.)
- You are required to use Dijkstra's Shortest Path algorithm when you flee. Your output should be the list of rooms, starting with the room you are in when you flee, and the directions taken through each room along your path.
- When something is taken from the knapsack by a hag, you can use whatever user interface you want that is clear to show and select the item to give her.
- When something is stolen by the hunchback, the user does not get to chose. Explain how you do this in your documentation/design. You could take the first thing, you could generate a random number, etc. Tell the user what was stolen.
- When the user gives the U (Use) command, you can use whatever interface you want that is clear to select the item. Then tell the user how the situation has changed.
- Your program output should be neat, clear and understandable.
- I will run this from the java command line or JGrasp. JGrasp can handle applets (yes, even reading files without security problems) and frames. Let me know if you have run preferences in your email submission.
- Documentation is critical.
- You must submit a design document describing your algorithms, approach, and class file layout. This should include your javadoc files.
- Your electronic submission must only include source files.
- Your hardcopy submission should be in a small binder with tabs or some other type of organizational form.