1. As instructed, the cost of moving through each terrain is based on guessing how quickly i could walk through that terrain, using open land as the baseline i represented these values as a percentage, 1.5 for example representing i would take 50% longer to walk through a terrain.
2. My cost function is the 2 dimensional distance, plus the change in elevation if it is an uphill movement, all multiplied by the terrain modifier. My heuristic is identical, but without the terrain modifier. Since the minimum value for that modifier is 1.0, my heuristic is admissible.
3. The leafify function used for fall events simply looks through the entire image, and any footpath with an adjacent easy movement forest becomes leafy.
4. The freeze function used for winter events has 2 parts. In the first, it scans the whole image to find water tiles that are directly adjacent to land. In the second part, it uses a queue to perform a BFS starting from the list formed in the first part.
5. The flood function behaves much like freeze, except it adds the land tiles that are adjacent to the water tiles, and also records the water level of the tile that each land tile was adjacent to. The second part carries this elevation value out to the end.
6. The printPath function colors all tiles in the calculated path one color, and those tiles neighbors in another. Finally, the neighbors of each goal location are given a third new color to stand out.

A note: my program assumes the season is given in lowercase letters as “winter”, “fall”, or “spring”. Any other input for this field will result in a summer map being used.