CS142, Programming II **Due: 23:59, Tuesday 02/28/2023**

Name: Majd Kawak / NetID: mkawa025 / CodeForces ID: mkawa025 / Student ID: 862273310

A. Wizard Chess

Submission#: 195386956

In this program, I start off with allocating an Int to act as board counter to solve multiple boards, $counter$ gets increased every time a board gets solved. Then I go and create an int to get the number of spots taken $spots\ Taken$, then I create a 2D $vector$ to model my board. After taking input, I marked my $Start$ and $Dest$ position on the board to solve for minimum steps to reach destination. Since we don't have weights or price to move our $Knight$ around, I used BFS to find the solution for this problem.

C. Making New Friends

Submission#: 195383073

I tried Kruskal's algorithm with union sets to result in minimum spanning tree for the class but failed, got partial point 0.3 :(

E. Add Oil

Submission#: 195398744

In this program, firstly I declared an int that would hold number of cities is in the map(graph), an int for roads(edges) number, an int for minimum gas tank capacity $min\_Capacity$ and a $pair$ of 3 $ints$ to represent edges $< value\_Distance, city_1, city_2 >$.
After inputing all the above, I $sort$ all road based on their $value\_Distance$ from shortest to largest. Then I declare a $vector\ root$ to hold each city(vertex) parent. At first, all cities root are initialized to itself (Meaning root of city i when 3 is 3).

Computing $min\_Capacity$ using Kruskal's Algorithm. Using a for loop, I iterate through all roads starting from shortest to largest. At each iteration I check if parent of city u is the same for city v using $parent$ function, if not; I do $union\_Set$ operation for u and v update $root$ vector, assign $route\_Distance$ to $min\_Capacity$ and continue.

Lastly, after iterating through all roads(edges) I end up with $min\_Capacity$ pointing to last route picked from roads to span our map cities, this would represent minimum gas tank capacity to travel form city x to y which is the longest route.

I implemented Kruskal's Algorithm using $union\_find$, Total time complexity = $O(n^2 + m\ log\ n)$.

F. Selling Candies

Submission#: 195398766

In this program, firstly I declared an int that would hold number of cities is in the map(graph), an int for number of roads(edges) number, an array of ints to hold selling candies value for each city $candy_Profit$, a list of adjacency list to model our map(Vertices and Edges) and a inverted adjacency list to model roads

back to Riverside(source vertex 0).
After inputing all the above, I declare a function $compute_Dijkstra$ that would take a $list$, source vertex s, number of cities in graph num_Cities and an $array$ to hold our computed distances from source vertex to all nodes $distance_From_Source$.
$compute_Dijkstra$ follows Dijkstra's algorithm implementing binary heap as priority queue to compute $SSSP$ from source to all other nodes. After computing all tenitave distances from vertex 0(Riverside) to all other cities, we pass inverted list, source vertex and another array to compute $SSSP$ from all nodes back to Riverside(vertex 0).
Last step is to take 2 $array$s computed and subtract each indices sum from $candy_Profit$ array at each index. Store computed result in $candy_Profit$ at each index, finally display max element in $candy_Profit$ $array$ that would represent max_Profit from all cities including Riverside(vertex 0). Total time complexity $= O(m + n\ log\ n) + O(n) = O(m + n\ log\ n)$.