```
/* Target Logo Image gotten from Google Images at
https://www.pinterest.com/pin/268456827775188206
*/
// Longitude and latitudes used in test cases gotten from Google Maps
/*  Import data columns from Studio code library
    Store as lists
*/
var addressList = getColumn("Target Store Locations", "Address"); //  Target address column
var cityList = getColumn("Target Store Locations", "City"); //  Target city column
var latitudeList = getColumn("Target Store Locations", "Latitude"); //  Target latitude column
var longitudeList = getColumn("Target Store Locations", "Longitude"); //  Target longitude colum
var postalcodeList = getColumn("Target Store Locations", "Postal Code"); //  Target postalcode
var stateList = getColumn("Target Store Locations", "State Code"); //  Target state column

/*  onEvent for Find Store button
    Gets two user inputs of latitude and longitude
    Calls distancesList function with parameters
    Changes screen and displays closest address using returned index

*/
onEvent("storeButton", "click", function( ) {
  // Get Inputs (latitude and longitude)
  var userLatitude = getText("latitudeInput");
  var userLongitude = getText("longitudeInput");

  //  Stores return of distancesList function in var index (int)
  var index = distancesList(userLatitude,userLongitude,latitudeList,longitudeList);

  // Displays output text box and hides image
  setProperty("outputText", "hidden", false);
  setProperty("targetImage", "hidden", true);
  if (index == -1) {
    // Prints that there is no nearby Target location
    setText("outputText", "\nThere is no nearby Target location within a 50 km radius");
    setProperty("outputText", "text-align", "center");
  } else {
    // Prints the address to the nearest target store accessing the index of address, city, stat
    setText("outputText", "\nThe address to the closest Target Store is: \n\n" +
    addressList[index] + "\n" + cityList[index] + ", " + stateList[index] + " " + postalcodeList
    setProperty("outputText", "text-align", "center");
  }
});

/*  distancesList function
    parameters: int (user inputted latitude), int (user inputted longitude),
                list (target Latitudes), list (target Longitudes)
    returns: int (index of target location closest to inputted latitude and longitude)
    Uses parts of haversine function from the internet
      haversine is a function that finds the distance, in kilometers, between two latitude/longi
      Gotten from https://pypi.org/project/haversine/
*/
function distancesList(lat,lon,latitudeList,longitudeList) {

```

```
52    // Create a list for all distances from inputted location
53    var distances = [];
54
55    // Convert degrees to radians
56    // Degrees to radians constant gotten from Google Calculator
57    var lat1 = lat * 0.0174533;
58    var lon1 = lon * 0.0174533;
59    var radius = 6371;
60
61    // Iterate through list to first convert each latitude and longitude to radians
62    // Then find distances using Haversine function
63    for (var i = 0; i < latitudeList.length; i++) {
64    // Convert each latitude and longitude to radians
65      var lat2 = latitudeList[i] * 0.0174533;
66      var lon2 = longitudeList[i] * 0.0174533;
67      var dlat = lat2 - lat1;
68      var dlon = lon2 - lon1;
69      var a = Math.pow(Math.sin(dlat / 2),2) + Math.cos(lat1) * Math.cos(lat2) * Math.pow(Math.sir
70      var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
71      var distance = radius * c;
72      appendItem(distances, distance);
73      // Uses sequencing to get correct distance between locations, stores in a list
74    }
75
76    var minDistanceIndex = 0;
77    // Iterate through list of distances
78    for (var x = 0; x < distances.length; x++) {
79      // Use selection to find shortest distance
80      if (distances[x] < distances[minDistanceIndex]) {
81        // Store shortest distance index (int)
82        minDistanceIndex = x;
83      }
84    }
85    // Selection to determine if there is a nearby Target location
86    if (distances[minDistanceIndex] > 50) {
87      // If distance is more than 50 km away, return -1, wont be in list
88      return -1;
89    }
90    else {
91      // Return index of shortest distance (int)
92      return minDistanceIndex;
93    }
94  }
95
```

PDF document made with CodePrint using Prism