Project code

MpPoints.swift:

```
//
//  MapPoints.swift
//  Never Get Lost
//
//  Created by Yusra Khalid on 1/7/18.
//  Copyright © 2018 Yusra Khalid. All rights reserved.
//

import Foundation
import MapKit

public class MapPoints {
    let points: [MKMapPoint];
    let links:[[Int ]]

    init() {
        self.points = [MKMapPoint(x:33.649238, y:72.999726),MKMapPoint(x:33.646792, y:
72.995162),MKMapPoint(x:33.644155, y:72.997061),MKMapPoint(x:33.644280, y:
72.997434),MKMapPoint(x:33.643782, y:72.997763),MKMapPoint(x:33.643284, y:
72.998149),MKMapPoint(x:33.643025, y:72.998381),MKMapPoint(x:33.642811, y:
72.998515),MKMapPoint(x:33.642288, y:72.998891),MKMapPoint(x:33.641917, y:
72.999159),MKMapPoint(x:33.642533, y:73.000345),MKMapPoint(x:33.643507, y:
72.999503),MKMapPoint(x:33.643443, y:73.002089),MKMapPoint(x:33.646461, y:
73.001544),MKMapPoint(x:33.643051, y:72.996498),MKMapPoint(x:33.642578, y:
72.996777),MKMapPoint(x:33.642087, y:72.997153),MKMapPoint(x:33.641569, y:
72.997561),MKMapPoint(x:33.641221, y:72.997818),MKMapPoint(x:33.641739, y:
73.001021),MKMapPoint(x:33.641123, y:73.000031),MKMapPoint(x:33.636889, y:
72.989614),MKMapPoint(x:33.634906, y:72.991116),MKMapPoint(x:33.635808, y:
72.987479),MKMapPoint(x:33.639435, y:72.987661),MKMapPoint(x:33.640435, y:
72.989099),MKMapPoint(x:33.642052, y:72.989335),MKMapPoint(x:33.642150, y:
72.985687),MKMapPoint(x:33.641060, y:72.983681),MKMapPoint(x:33.642734, y:
72.987658),MKMapPoint(x:33.642582, y:72.988249),MKMapPoint(x:33.642966, y:
72.988807),MKMapPoint(x:33.643547, y:72.989161),MKMapPoint(x:33.643842, y:
72.988110),MKMapPoint(x:33.644221, y:72.986559),MKMapPoint(x:33.644493, y:
72.986471),MKMapPoint(x:33.644368, y:72.985167),MKMapPoint(x:33.645217, y:
72.985424),MKMapPoint(x:33.643706, y:72.985103),MKMapPoint(x:33.642536, y:
72.982619),MKMapPoint(x:33.643786, y:72.986562),MKMapPoint(x:33.645550, y:
72.985205),MKMapPoint(x:33.645970, y:72.980238),MKMapPoint(x:33.645246, y:
72.985873),MKMapPoint(x:33.645532, y:72.986028),MKMapPoint(x:33.644912, y:
72.987604),MKMapPoint(x:33.645716, y:72.988044),MKMapPoint(x:33.644537, y:
72.988634),MKMapPoint(x:33.645305, y:72.989063),MKMapPoint(x:33.646404, y:
72.986349),MKMapPoint(x:33.644814, y:72.990490),MKMapPoint(x:33.645537, y:
72.990941),MKMapPoint(x:33.646126, y:72.989546),MKMapPoint(x:33.647189, y:
72.990072),MKMapPoint(x:33.647207, y:72.988055),MKMapPoint(x:33.646894, y:
72.991177),MKMapPoint(x:33.644750, y:72.992454),MKMapPoint(x:33.645759, y:
72.993023),MKMapPoint(x:33.643249, y:72.994203),MKMapPoint(x:33.642204, y:
72.993752),MKMapPoint(x:33.641284, y:72.991832),MKMapPoint(x:33.648752, y:
72.993106),MKMapPoint(x:35.000345, y:80.354366)]
```

```
        self.links = [[0,1],[1,0,2,57],[2,1,3,58],[3,2,4,13],[4,3,5,14],[5,4,6,15],[6,5,7,11],[7,6,8,16],
[8,11,7,9,17],[9,8,10,18],[10,9,11,12,19],[11,6,10,8],[12,10],[13,3],[14,4,15],[15,14,5,16],
[16,15,7,17],[17,16,8,18],[18,17,9,21],[19,10,20],[20,19],[21,18,22,23,24],[22,21],[23,21],
[24,21,25,27],[25,24,26],[26,25,30,31,60],[27,24,28,29,38],[28,27,39],[29,27,30],[30,29,31,26],
[31,30,26,32],[32,31,33],[33,32,34],[34,33,35],[35,34,36],[36,35,38,37],[37,36,43,41],
[38,39,40,27,36],[39,38,28],[40,38],[41,37,42,44,49],[42,41],[43,37,44],[44,41,43,49],[45,46],
[46,45,48,49],[47,48],[48,47,46,50,52],[49,44,46,54,41],[50,48,51],[51,50,56,52],[52,51,48,53],
[53,52,54,55],[54,53,49],[55,53,61,57],[56,57,51],[57,56,55,1,58],[58,57,59,2],[59,58,60],
[60,59,26],[61,55],[62]]
    }
}
```

Vertex.swift:

```
//
//  Vertex.swift
//  Never Get Lost
//
//  Created by Yusra Khalid on 1/7/18.
//  Copyright © 2018 Yusra Khalid. All rights reserved.
//

import Foundation
import MapKit

class Vertex{
    var name: Int;
    var h: Double; // distance between selected vertex and destination
    var g: Double; //distance between last vertex to this selected vertex
    var f: Double;
    var parent:Int
    var destination:Int;
    var p = MapPoints() // creating object of mappoints to access points of graph from other
class

//    constructor to initilize object with destination and parent
//    value of g,h,f are set as 0. it will change on update
    init(name:Int, destination:Int, parent:Int) {
        self.destination = destination
        self.name = name
        self.g = 0.0
        self.h = 0.0
        self.f = g+h
        self.parent = parent
    }

    func distance(other: Vertex) -> Double {
      return MKMetersBetweenMapPoints(p.points[self.name], p.points[other.name])
    }
```

```
//    on updating the parent of a vertex it's g, h, and f value will be updated
    func update(parent:Int, gScore: Double) {
        self.parent = parent
        self.g = MKMetersBetweenMapPoints(p.points[self.name], p.points[parent]) + gScore
        self.h = MKMetersBetweenMapPoints(p.points[self.name], p.points[destination])
        self.f = g+h
    }
}
```

AAsterik.swift:

```
//
//  AAsterik.swift
//  Never Get Lost
//
//  Created by Yusra Khalid on 1/7/18.
//  Copyright © 2018 Yusra Khalid. All rights reserved.
//

import Foundation
import MapKit

class AAsterik{
    var start:Int
    var end:Int
    var open:[Int] = []
    var closed:[Int] = []
    var v = Vertex(name: -1, destination: -1, parent: -1) // creating object of vertex to access it's
info
    var vertices:[Vertex] = []
    var gScore: [Double] = []

    init(start:Int, end:Int){
        self.start = start
        self.end = end
        open = []
        closed = []
        for i in 0...v.p.points.count - 1{
            let v = Vertex(name: i, destination: self.end, parent: -1)
            vertices.append(v)
            gScore.append(9999999)
        }

    }

    func getLowestNode() -> Int {
        var min:Int = open[0]
        var index:Int = 0
        for n in 0...open.count - 1 {
            if vertices[open[n]].f < vertices[min].f {
                min = open[n];
```

```
            index = n
        }
    }
    open.remove(at: index)
    return min
}

func getPath() -> [Int] {
    var temp:Int = end
    var path:[Int] = []

    path.append(end)
    while (temp != start){
        path.append(vertices[temp].parent)
        temp = vertices[temp].parent
    }
    return path

}

func pathFinding() -> [Int] {
    var current = self.start

    gScore[start] = 0
    open.append(start)

    while (!open.isEmpty){
        current = getLowestNode()

        if (current == end){
            return getPath()
        }

        closed.append(current)

        for neighbour in v.p.links[current]{
            var flag = true
            for k in open{
                if (k == neighbour){
                    flag = false
                }
            }
            for l in closed {
                if (l == neighbour){
                    flag = false
                }
            }
            if (flag){
                open.append(neighbour)
            }

            // The distance from start to a neighbor
            //the "dist_between" function may vary as per the solution requirements.
```

```
            let tentative_gScore = gScore[current] + vertices[current].distance(other:
vertices[neighbour])
            if tentative_gScore >= gScore[neighbour] {
                continue        // This is not a better path.
            }

            // This path is the best until now. Record it!
            vertices[neighbour].update(parent: current, gScore: gScore[current])
            gScore[neighbour] = tentative_gScore
        }
    }
    return []
  }
}
```

ArtWork.swift:

```
//
//  Artwork.swift
//  Never Get Lost
//
//  Created by Yusra Khalid on 1/7/18.
//  Copyright © 2018 Yusra Khalid. All rights reserved.
//

import Foundation
import MapKit

class Artwork: NSObject, MKAnnotation {
  let title: String?
  let coordinate: CLLocationCoordinate2D

  init(title: String, coordinate: CLLocationCoordinate2D) {
    self.title = title
    self.coordinate = coordinate
    super.init()
  }

  var subtitle: String? {
    return title
  }
}
```

MapViewCustomDelegate.swift:

```
//
//  MapViewCustomDelegate.swift
//  Never Get Lost
//
//  Created by Yusra Khalid on 1/7/18.
//  Copyright © 2018 Yusra Khalid. All rights reserved.
```

```
//

import Foundation
import MapKit

class MapViewCustomDelegate: NSObject, MKMapViewDelegate {

    var start: Int = -1
    var end: Int = -1
    var startSelected: Bool = false
    let maps = MapPoints()
    weak var mapView: MKMapView!
    weak var viewController: ViewController?

    init(mapView: MKMapView, viewController: ViewController?) {
        self.mapView = mapView
        self.viewController = viewController
        super.init()
    }

    func resetDefault() {
        start = -1
        end = -1
        startSelected = false
    }

    func findPath() {
        let d = AAsterik(start: self.start, end: self.end)
        let p = d.pathFinding()
        drawPath(points: p)
    }

    func drawPath(points: [Int]){

        var locations = points.map {
            CLLocationCoordinate2D(latitude: self.maps.points[$0].x, longitude:
self.maps.points[$0].y)
        }
        let polyline = MKPolyline(coordinates: &locations, count: locations.count)
        mapView.add(polyline)
    }

    func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) ->
MKOverlayRenderer {
        print ("overlay function called")
        if overlay is MKPolyline {
            let polylineRenderer = MKPolylineRenderer(overlay: overlay)
            polylineRenderer.strokeColor = UIColor.blue
            polylineRenderer.lineWidth = 5
            return polylineRenderer
        }
        return MKOverlayRenderer()
    }
```

```
    func mapView(_ mapView: MKMapView, didSelect: MKAnnotationView) {

        guard let annotation = didSelect.annotation as? Artwork else { return }
        let id = Int(annotation.title!)!

        var message = ""
        var buttonText = ""

        if startSelected {
            message = "Select current position as EndPoint?"
            buttonText = "Endpoint"
        } else {
            message = "Select current position as StartPoint?"
            buttonText = "StartPoint"
        }

        //Creating UIAlertController and
        //Setting title and message for the alert dialog
        let alertController = UIAlertController(title: "Select Location?", message: message,
preferredStyle: .alert)

        //the confirm action taking the inputs
        let confirmAction = UIAlertAction(title: buttonText, style: .default) { (_) in

            if self.startSelected {
                self.end = id
                self.findPath()

                self.resetDefault()
            } else {
                self.startSelected = true
                self.start = id
            }
        }

        //the cancel action doing nothing
        let cancelAction = UIAlertAction(title: "Cancel", style: .cancel) { (_) in }

        //adding the action to dialogbox
        alertController.addAction(confirmAction)
        alertController.addAction(cancelAction)

        //finally presenting the dialog box
        viewController?.present(alertController, animated: true, completion: nil)
    }
}
```

ViewController.swift:

```
//
//  ViewController.swift
//  Never Get Lost
```

```
//
//  Created by Yusra Khalid on 1/7/18.
//  Copyright © 2018 Yusra Khalid. All rights reserved.
//

import UIKit
import MapKit

class ViewController: UIViewController {
    @IBOutlet weak var mapView: MKMapView!
    let maps = MapPoints()
    var customDelegate: MapViewCustomDelegate?


    override func viewDidLoad() {
        super.viewDidLoad()
        let initialLocation = CLLocation(latitude: 33.643786, longitude: 72.986562);
        customDelegate = MapViewCustomDelegate(mapView: mapView, viewController: self)
        mapView.delegate = customDelegate

        for i in 0...maps.points.count - 1 {
            let artwork = Artwork(title: String(i),
                                  coordinate: CLLocationCoordinate2D(latitude: maps.points[i].x, longitude:
maps.points[i].y))
            self.mapView.addAnnotation(artwork)
        }

        let regionRadius: CLLocationDistance = 2000
        func centerMapOnLocation(location: CLLocation) {
            let coordinateRegion = MKCoordinateRegionMakeWithDistance(location.coordinate,
                                                                      regionRadius, regionRadius)
            mapView.setRegion(coordinateRegion, animated: true)
        }

        centerMapOnLocation(location: initialLocation)

//        every possible path has been tested
//        for i in 0...61{
//            for j in 0...61{
//                if (i != j){
//                    let d = AAsterik(start: i, end: j)
//                    var q = d.pathFinding()
//                    for v in q{
//                        print(v , "\n")
//                    }
//                    print("\n \n \n")
//                }
//            }
//        }
    }
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
```

}