Question 3 and Question 4:
Implementation of undirectedGraph class, directedGraph class, GraphSearch class , and TopSort class can be found in **Graph.h**

All other functions required for creating undirected and directed graphs and a linked list, and the code required for testing can be found in **main.cpp**

Question 5:
Implementation of all classes and functions required in question 5; WeightedGraph class, creating complete weighted graph functions, Dijkstra's function and the code for testing, can be found in **Q5_Dijkstras.cpp**

Question 6:
Implementation of all classes and functions required in question 6 ; GridGraph class, creating grid Graph function, Astar function and the code for testing, can be found in **Q6_Astar.cpp**

Question 3:

Testing creating the graphs:

```
Adj lists of random unweighted graph of size 10 nodes:
0 :{2,3,4,6,7,9,}
1 :{2,4,6,7,9,}
2 :{0,1,3,7,}
3 :{0,2,6,7,9,}
4 :{0,1,5,8,9,}
5 :{4,6,7,8,9,}
6 :{0,1,3,5,9,}
7 :{0,1,2,3,5,}
8 :{4,5,}
9 :{0,1,3,4,5,6,}

Adj lists of linked list graph of size 10 nodes:
0 :{1,}
1 :{0,2,}
2 :{1,3,}
3 :{2,4,}
4 :{3,5,}
5 :{4,6,}
6 :{5,7,}
7 :{6,8,}
8 :{7,9,}
9 :{8,}

Program ended with exit code: 0
```

Testing DFS and BFT:

```
Adj lists of random unweighted graph of size 7 nodes:
0 :{2,3,}
1 :{3,}
2 :{0,6,}
3 :{0,1,5,6,}
4 :{}
5 :{3,6,}
6 :{2,3,5,}

DFS Rec from 0 to 6 output:
0, 2, 6,
DFS Iter from 0 to 6 output:
0, 3, 6,
BFT Rec output:
0, 2, 3, 6, 1, 5, 4,
BFT Iter output:
0, 2, 3, 6, 1, 5, 4,
Program ended with exit code: 0
```

Testing removal of undirected edges:

1- Removing edge between 0 and 5:

```
Adj lists of random unweighted graph of size 7 nodes:
0 :{4,5,6,}
1 :{3,4,5,}
2 :{4,5,6,}
3 :{1,5,}
4 :{0,1,2,6,}
5 :{0,1,2,3,}
6 :{0,2,4,}

Removing edge between 0 and 5
Adj lists after removal:
0 :{4,6,}
1 :{3,4,5,}
2 :{4,5,6,}
3 :{1,5,}
4 :{0,1,2,6,}
5 :{1,2,3,}
6 :{0,2,4,}
```

Testing BFT on a linked list size 10,000:

1- Output of the first few lines of BFT Iteratively:

```
BFT Iter on 10,000 size graph:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
    31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
    59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,
    87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111,
    112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
    134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
    156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
    178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199,
    200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
    222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243,
```

2- Output of the first few lines of BFT Recursively:

```
BFT Rec on 10,000 size graph:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
    31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
    59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,
    87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111,
    112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
    134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
    156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177,
    178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199,
    200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
    222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243,
```

Question 4:

Sample output

1- First few lines of the result of running Khans sort on a graph of size 1000:

```
Result of running Khans sort on Directed Graph:
1 0 4 5 8 2 3 10 6 9 7 12 13 11 15 14 16 18 17 21 23 20 19 25 22 24 27 28 29 30 26 31 32 33 35 36 37 34 38 39 40 41 44 43
    42 45 49 48 47 46 50 51 54 55 52 58 53 61 56 59 62 57 66 63 65 60 64 68 69 70 67 72 71 75 74 73 79 76 78 77 81 80 83
    82 84 85 86 88 87 93 89 90 91 94 92 95 97 96 100 99 105 98 102 101 104 103 108 106 111 107 109 112 113 117 110 121
    118 114 115 116 120 119 124 122 123 125 126 127 128 130 129 131 132 133 134 137 138 135 139 143 136 142 145 146 140
    144 141 148 147 149 150 154 151 158 155 152 156 153 159 160 157 161 162 163 166 171 164 167 165 168 169 170 173 174
    172 175 176 178 177 180 179 181 182 184 183 185 186 189 194 187 190 191 198 192 188 195 193 196 197 200 199 201 204
    202 203 209 205 207 210 206 213 208 211 212 214 216 215 217 218 219 221 224 225 220 222 223 226 228 227 229 230 231
    233 232 234 235 237 238 236 240 242 244 239 243 248 245 246 250 247 241 249 251 254 253 255 252 256 258 257 259 261
    263 262 260 264 265 267 266 268 272 274 269 271 273 270 275 276 278 277 281 279 282 280 283 284 286 285 287 288 292
    289 290 291 297 294 293 298 296 299 295 301 300 305 302 304 303 307 306 311 309 308 314 315 310 312 316 313 317 318
    319 320 321 322 323 325 327 324 326 333 328 332 329 331 336 334 335 330 341 338 339 340 337 342 343 344 349 346 345
```

2- First few lines of the result of running mDFS sort on a graph of size 1000:

```
Result of running mDFS sort on Directed Graph:
1 0 4 5 8 2 10 3 6 9 12 7 13 11 15 14 16 18 21 17 23 20 25 19 22 27 28 24 30 29 26 31 32 33 35 36 37 34 38 39 40 41 44 43
    42 45 49 48 47 50 46 51 55 54 52 58 53 61 56 59 62 57 66 63 65 60 64 69 68 70 67 72 71 75 74 73 79 76 78 77 81 80 83
    82 84 85 86 88 93 87 89 90 91 94 92 95 97 96 100 105 98 101 99 102 104 108 111 107 103 109 112 106 113 117 121 110
    118 114 115 116 120 119 124 122 123 126 125 127 128 130 131 129 132 134 133 137 138 135 139 143 142 136 145 146 140
    144 148 149 141 147 150 154 151 158 155 152 156 153 159 160 161 157 162 163 166 171 164 167 165 168 169 170 173 174
    175 172 176 178 177 180 179 181 184 182 183 185 189 186 194 187 190 191 198 192 188 195 193 196 197 200 199 204 201
    202 209 205 203 207 210 206 213 208 211 212 214 216 215 217 218 219 221 224 225 220 222 223 226 228 227 229 230 231
    233 232 234 235 238 237 236 240 242 244 239 248 243 246 245 250 247 241 249 251 254 253 255 256 252 258 257 259 261
    263 262 260 264 267 266 265 268 274 272 269 271 273 270 275 276 278 277 281 279 282 280 283 286 284 285 287 288 292
    289 290 291 297 294 293 298 296 299 295 301 300 305 302 304 307 303 306 311 309 315 314 308 312 316 313 317 310 318
    319 320 321 322 323 325 327 324 326 333 328 332 329 331 336 334 335 330 341 338 339 340 337 342 343 344 349 346 345
```

Question 5:

Sample output

Creating a random graph of size 10 nodes and running Dijkstra's on it.

```
Random weighted Graph:
0 :{(4,7), (9,6), (5,2), (6,10), (7,7), (8,5), (2,8), (3,7), (1,2), }
1 :{(4,8), (9,5), (5,2), (6,7), (7,5), (8,7), (2,3), (3,6), (0,1), }
2 :{(4,5), (9,4), (5,9), (6,3), (7,3), (8,1), (3,5), (1,1), (0,6), }
3 :{(4,5), (9,1), (5,5), (6,3), (7,3), (8,3), (2,6), (1,3), (0,9), }
4 :{(9,7), (5,4), (6,9), (7,4), (8,10), (2,4), (3,9), (1,7), (0,10), }
5 :{(4,10), (9,7), (6,5), (7,7), (8,6), (2,2), (3,5), (1,6), (0,8), }
6 :{(4,5), (9,1), (5,3), (7,9), (8,8), (2,8), (3,9), (1,6), (0,1), }
7 :{(4,1), (9,7), (5,8), (6,7), (8,5), (2,6), (3,8), (1,5), (0,2), }
8 :{(4,3), (9,7), (5,3), (6,6), (7,8), (2,1), (3,1), (1,2), (0,9), }
9 :{(4,9), (5,10), (6,4), (7,6), (8,4), (2,5), (3,7), (1,1), (0,10), }
Minimum distance from start to each node using Dijkstra's Alg. :
4: 7
9: 6
5: 2
6: 7
7: 7
8: 5
2: 4
3: 6
1: 2
0: 0
```

Creating a linked list of size 10 nodes and running Dijkstra's on it.

```
Linked List:
0 :{(1,1), }
1 :{(2,1), }
2 :{(3,1), }
3 :{(4,1), }
4 :{(5,1), }
5 :{(6,1), }
6 :{(7,1), }
7 :{(8,1), }
8 :{(9,1), }
9 :{}
Minimum distance from start to each node using Dijkstra's Alg. :
1: 1
2: 2
4: 4
9: 9
5: 5
6: 6
7: 7
8: 8
0: 0
3: 3
Program ended with exit code: 0
```

Question 6:

Sample output

Creating a random graph of size 5*5.

```
GridGraph of size 5 * 5:
0 :{1,5,}
1 :{0,2,6,}
2 :{1,3,7,}
3 :{2,4,8,}
4 :{3,}
5 :{0,10,}
6 :{1,11,}
7 :{2,8,}
8 :{3,7,9,13,}
9 :{8,14,}
10 :{5,11,15,}
11 :{6,10,16,}
12 :{17,}
13 :{8,14,18,}
14 :{19,9,13,}
15 :{20,10,}
16 :{11,21,}
17 :{12,22,}
18 :{23,13,}
19 :{14,}
20 :{15,21,}
21 :{16,20,22,}
22 :{17,21,}
23 :{18,24,}
24 :{23,}
```

```
Minimum distance from start node using A* Algorithm to find minimum path from node (0,0), to (24,24):
0:    d[v]:0
1:    d[v]:1
9:    d[v]:5
5:    d[v]:1
6:    d[v]:2
2:    d[v]:2
3:    d[v]:3
7:    d[v]:3
8:    d[v]:4
10:    d[v]:2
11:    d[v]:3
13:    d[v]:5
14:    d[v]:6
15:    d[v]:3
16:    d[v]:4
17:    d[v]:7
18:    d[v]:6
20:    d[v]:4
21:    d[v]:5
22:    d[v]:6
23:    d[v]:7
24:    d[v]:8
Program ended with exit code: 0
```