DMDD Final Project Simulation
Zixiao 20200328

Simulation

Content

Abstract

In this simulation, we didn't consider multithreading, which means we assume there is only one cumstomer is using the application.

So the main goal for this simulatio is to test if the database can store the data and interact with each other correctly.

The database for this simulation is as following:

This simulation include following things:

- 1. Test if the database can store right data into tables:
  - Map data simulation
    - `building`
    - `floorMap`
    - `floorMapPoints`
    - `locationToVisit`
    - `locationToChangeFloor`
    - `beacons`
  - User information simulation
    - `adminUser`
    - `customer`
- 2. Test if the connections between device and beacon are generated correctly
  - `device`
  - `deviceBeaconConnection`
- 3. Test if the positioning is updated correctly
  - `deviceCurrentPosition`
- 4. Test if the navigation is generated correctly
  - `navigationRequest`
- 5. Test if the paths are generated correctly
  - `singleFloorPath`
  - `singleFloorPathPoints`
  - `floorAlongtheWay`
  - `beginFloor`
  - `midFloor`
  - `endFloor`
  - `midFloorChangeCombination`
  - `pathCombination`
- 6. Test if the system is update correctly when navigation request is over
  - `navigationComplete`
  - `navigationRequest`

Connect to database

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
# import the magic code for using sql in jupyter notebook
%load_ext sql
```

```
# local database
# %sql mysql+pymysql://root:fjwwzx970814@localhost/mydb
# remote database
%sql mysql+pymysql://brickea_mac:fjwWZX970814@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
```

```
'Connected: brickea_mac@mydb'
```

```
%%sql
show tables
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
21 rows affected.
```

Tables_in_mydb
adminuser
beacons
beginfloor
building
customer
device
devicebeaconconnection
devicecurrentposition
endfloor
flooralongtheway
floormap
floormappoints
locationtochangefloor
locationtovisit
midfloor
midfloorchangecombination
navigationcomplete
navigationrequest
pathcombination
singlefloorpath
singlefloorpathpoints

```
import sqlalchemy as sqlManager
```

```
# Create connection with database
connection = sqlManager.create_engine('mysql+pymysql://brickea_mac:fjwWZX970814@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb?charset=utf8')
```

Map and user data simulation and test

Map data simulation

Because I only got one map sample as following

So I will use it as a template to generate 5 different floor maps for 2 different buildings

And I assume the coordinate original in the map is at its top left

The map is 20 X 21

Here I assume

- points within green line is a wall
- points within red circle is an elevator
- points within purple circle is a stair
- elevator and stair are in the same position for every floor
- elevator and stair can help human move to any floor

building - table test

```
# Generate building data
building = pd.DataFrame([
    [1,'building_1','address_1','description'],
    [2,'building_2','address_2','description'],
],columns=['idbuilding','name','address','description'])
building
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	idbuilding	name	address	description
0	1	building_1	address_1	description
1	2	building_2	address_2	description

```
# Insert data into database
pd.io.sql.to_sql(building,'building',connection,schema='mydb',if_exists='append',index=False)
```

```
/Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py:170: Warning: (3719,
"'utf8' is currently an alias for the character set UTF8MB3, but will be an alias for UTF8MB4 in a
future release. Please consider using UTF8MB4 in order to be unambiguous.")
  result = self._query(query)



---------------------------------------------------------------------------

IntegrityError                            Traceback (most recent call last)
```

```
~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self,
dialect, constructor, statement, parameters, *args)
   1228                     self.dialect.do_executemany(
-> 1229                         cursor, statement, parameters, context
   1230                     )


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/dialects/mysql/mysqldb.py in
do_executemany(self, cursor, statement, parameters, context)
    147     def do_executemany(self, cursor, statement, parameters, context=None):
--> 148         rowcount = cursor.executemany(statement, parameters)
    149         if context is not None:


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in executemany(self, query, args)
    196                                           self.max_stmt_length,
--> 197                                           self._get_db().encoding)
    198


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in _do_execute_many(self, prefix,
values, postfix, args, max_stmt_length, encoding)
    233             sql += v
--> 234         rows += self.execute(sql + postfix)
    235         self.rowcount = rows


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    169
--> 170         result = self._query(query)
    171         self._executed = query


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in _query(self, q)
    327         self._clear_result()
--> 328         conn.query(q)
    329         self._do_get_result()


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in query(self, sql, unbuffered)
    516         self._execute_command(COMMAND.COM_QUERY, sql)
--> 517         self._affected_rows = self._read_query_result(unbuffered=unbuffered)
    518         return self._affected_rows


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_query_result(self,
unbuffered)
    731             result = MySQLResult(self)
--> 732             result.read()
    733         self._result = result


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in read(self)
   1074         try:
-> 1075             first_packet = self.connection._read_packet()
   1076


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_packet(self,
packet_type)
    683         packet = packet_type(buff, self.encoding)
--> 684         packet.check_error()
    685         return packet


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/protocol.py in check_error(self)
    219             if DEBUG: print("errno =", errno)
```

```
--> 220             err.raise_mysql_exception(self._data)
    221


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/err.py in raise_mysql_exception(data)
    108     errorclass = error_map.get(errno, InternalError)
--> 109     raise errorclass(errno, errval)


IntegrityError: (1062, "Duplicate entry '1' for key 'PRIMARY'")


The above exception was the direct cause of the following exception:


IntegrityError                            Traceback (most recent call last)

<ipython-input-6-fbfdf4ff6e62> in <module>
      1 # Insert data into database
----> 2
pd.io.sql.to_sql(building,'building',connection,schema='mydb',if_exists='append',index=False)


~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/sql.py in to_sql(frame, name, con, schema,
if_exists, index, index_label, chunksize, dtype, method)
    516         chunksize=chunksize,
    517         dtype=dtype,
--> 518         method=method,
    519     )
    520


~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/sql.py in to_sql(self, frame, name, if_exists,
index, index_label, schema, chunksize, dtype, method)
   1318         )
   1319         table.create()
-> 1320         table.insert(chunksize, method=method)
   1321         if not name.isdigit() and not name.islower():
   1322             # check for potentially case sensitivity issues (GH7815)


~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/sql.py in insert(self, chunksize, method)
    754
    755                 chunk_iter = zip(*[arr[start_i:end_i] for arr in data_list])
--> 756                 exec_insert(conn, keys, chunk_iter)
    757
    758     def _query_iterator(


~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/sql.py in _execute_insert(self, conn, keys,
data_iter)
    668         """
    669         data = [dict(zip(keys, row)) for row in data_iter]
--> 670         conn.execute(self.table.insert(), data)
    671
    672     def _execute_insert_multi(self, conn, keys, data_iter):


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in execute(self, object_,
*multiparams, **params)
    986             raise exc.ObjectNotExecutableError(object_)
    987         else:
--> 988             return meth(self, multiparams, params)
    989
    990     def _execute_function(self, func, multiparams, params):
```

```
~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/sql/elements.py in
_execute_on_connection(self, connection, multiparams, params)
    285     def _execute_on_connection(self, connection, multiparams, params):
    286         if self.supports_execution:
--> 287             return connection._execute_clauseelement(self, multiparams, params)
    288         else:
    289             raise exc.ObjectNotExecutableError(self)


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in
_execute_clauseelement(self, elem, multiparams, params)
   1105             distilled_params,
   1106             compiled_sql,
-> 1107             distilled_params,
   1108         )
   1109         if self._has_events or self.engine._has_events:


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self,
dialect, constructor, statement, parameters, *args)
   1251         except BaseException as e:
   1252             self._handle_dbapi_exception(
-> 1253                 e, statement, parameters, cursor, context
   1254             )
   1255


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in
_handle_dbapi_exception(self, e, statement, parameters, cursor, context)
   1471                 util.raise_from_cause(newraise, exc_info)
   1472             elif should_wrap:
-> 1473                 util.raise_from_cause(sqlalchemy_exception, exc_info)
   1474             else:
   1475                 util.reraise(*exc_info)


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/util/compat.py in raise_from_cause(exception,
exc_info)
    396     exc_type, exc_value, exc_tb = exc_info
    397     cause = exc_value if exc_value is not exception else None
--> 398     reraise(type(exception), exception, tb=exc_tb, cause=cause)
    399
    400


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/util/compat.py in reraise(tp, value, tb,
cause)
    150             value.__cause__ = cause
    151         if value.__traceback__ is not tb:
--> 152             raise value.with_traceback(tb)
    153         raise value
    154


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self,
dialect, constructor, statement, parameters, *args)
   1227                 if not evt_handled:
   1228                     self.dialect.do_executemany(
-> 1229                         cursor, statement, parameters, context
   1230                     )
   1231             elif not parameters and context.no_parameters:


~/opt/anaconda3/lib/python3.7/site-packages/sqlalchemy/dialects/mysql/mysqldb.py in
do_executemany(self, cursor, statement, parameters, context)
    146
    147     def do_executemany(self, cursor, statement, parameters, context=None):
```

```
--> 148         rowcount = cursor.executemany(statement, parameters)
    149         if context is not None:
    150             context._rowcount = rowcount


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in executemany(self, query, args)
    195             return self._do_execute_many(q_prefix, q_values, q_postfix, args,
    196                                          self.max_stmt_length,
--> 197                                          self._get_db().encoding)
    198
    199         self.rowcount = sum(self.execute(query, arg) for arg in args)


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in _do_execute_many(self, prefix,
values, postfix, args, max_stmt_length, encoding)
    232                 sql += b','
    233             sql += v
--> 234         rows += self.execute(sql + postfix)
    235         self.rowcount = rows
    236         return rows


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    168         query = self.mogrify(query, args)
    169
--> 170         result = self._query(query)
    171         self._executed = query
    172         return result


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py in _query(self, q)
    326         self._last_executed = q
    327         self._clear_result()
--> 328         conn.query(q)
    329         self._do_get_result()
    330         return self.rowcount


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in query(self, sql, unbuffered)
    515                 sql = sql.encode(self.encoding, 'surrogateescape')
    516         self._execute_command(COMMAND.COM_QUERY, sql)
--> 517         self._affected_rows = self._read_query_result(unbuffered=unbuffered)
    518         return self._affected_rows
    519


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_query_result(self,
unbuffered)
    730         else:
    731             result = MySQLResult(self)
--> 732             result.read()
    733         self._result = result
    734         if result.server_status is not None:


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in read(self)
   1073     def read(self):
   1074         try:
-> 1075             first_packet = self.connection._read_packet()
   1076
   1077             if first_packet.is_ok_packet():


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/connections.py in _read_packet(self,
packet_type)
    682
    683         packet = packet_type(buff, self.encoding)
```

```
--> 684         packet.check_error()
    685         return packet
    686


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/protocol.py in check_error(self)
    218             errno = self.read_uint16()
    219             if DEBUG: print("errno =", errno)
--> 220             err.raise_mysql_exception(self._data)
    221
    222     def dump(self):


~/opt/anaconda3/lib/python3.7/site-packages/pymysql/err.py in raise_mysql_exception(data)
    107         errval = data[3:].decode('utf-8', 'replace')
    108     errorclass = error_map.get(errno, InternalError)
--> 109     raise errorclass(errno, errval)


IntegrityError: (pymysql.err.IntegrityError) (1062, "Duplicate entry '1' for key 'PRIMARY'")
[SQL: INSERT INTO mydb.building (idbuilding, name, address, description) VALUES (%(idbuilding)s, %
(name)s, %(address)s, %(description)s)]
[parameters: ({'idbuilding': 1, 'name': 'building_1', 'address': 'address_1', 'description':
'description'}, {'idbuilding': 2, 'name': 'building_2', 'address': 'address_2', 'description':
'description'})]
(Background on this error at: http://sqlalche.me/e/gkpj)
```

```
%%sql
select * from building
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
2 rows affected.
```

idbuilding	name	address	description
1	building_1	address_1	description
2	building_2	address_2	description

floorMap - table test

```
def generate_floor_info(floor_number=[]):
    result = []
    floor_id = 0
    for building in range(len(floor_number)):
        for floor in range(floor_number[building]):
            result.append([floor_id+1,building+1,floor+1,
('building_'+str(building+1)+'_floor_'+str(floor+1)),True])
            floor_id+=1
    return result
```

```
floor_data = generate_floor_info(floor_number=[3,2])
floor_data
```

```
[[1, 1, 1, 'building_1_floor_1', True],
 [2, 1, 2, 'building_1_floor_2', True],
```

```
 [3, 1, 3, 'building_1_floor_3', True],
 [4, 2, 1, 'building_2_floor_1', True],
 [5, 2, 2, 'building_2_floor_2', True]]
```

```
floor_columns = ['idfloorMap','idbuilding','floorNumber','excelMap','ifupdate']
```

```
# Generate floor data for each building
floor = pd.DataFrame(data=floor_data,columns=floor_columns)
floor
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	idfloorMap	idbuilding	floorNumber	excelMap	ifupdate
0	1	1	1	building_1_floor_1	True
1	2	1	2	building_1_floor_2	True
2	3	1	3	building_1_floor_3	True
3	4	2	1	building_2_floor_1	True
4	5	2	2	building_2_floor_2	True

```
# Insert data into database
pd.io.sql.to_sql(floor,'floorMap',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from floorMap
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
5 rows affected.
```

idfloorMap	idbuilding	floorNumber	excelMap	ifupdate
1	1	1	b'building_1_floor_1'	1
2	1	2	b'building_1_floor_2'	1
3	1	3	b'building_1_floor_3'	1
4	2	1	b'building_2_floor_1'	1
5	2	2	b'building_2_floor_2'	1

floorMapPoints - table test

Here I assume bottom left is the coordinate original point

```
np.zeros((3,3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
def generate_floor_basic_points(x_len,y_len):
    return np.zeros((x_len,y_len))
```

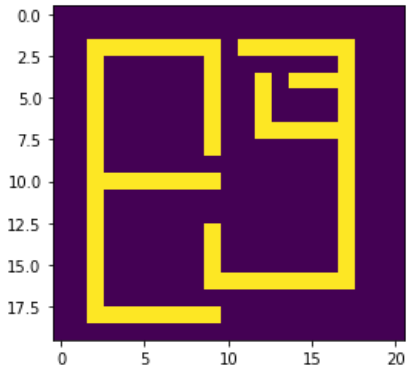
```
# The map is 20 X 21 (unit:meter)
floor_map_data = generate_floor_basic_points(20,21)
```

```
def generate_wall(floor_map,wall=[]):
    # wall should input coordinate of start point and end point
    # wall = [(start_x,start_y),(end_x,end_y)]
    start = wall[0]
    end = wall[1]
    for list_x in range(start[1],end[1]+1):
        for list_y in range(start[0],end[0]+1):
            floor_map[list_x][list_y] = 1
    return floor_map
```

```
# Generate wall
floor_map_data = generate_wall(floor_map_data,wall = [(2,2),(9,2)])
floor_map_data = generate_wall(floor_map_data,wall = [(9,2),(9,8)])
floor_map_data = generate_wall(floor_map_data,wall = [(2,2),(2,18)])
floor_map_data = generate_wall(floor_map_data,wall = [(2,18),(9,18)])
floor_map_data = generate_wall(floor_map_data,wall = [(2,10),(9,10)])
floor_map_data = generate_wall(floor_map_data,wall = [(9,13),(9,16)])
floor_map_data = generate_wall(floor_map_data,wall = [(9,16),(17,16)])
floor_map_data = generate_wall(floor_map_data,wall = [(11,2),(17,2)])
floor_map_data = generate_wall(floor_map_data,wall = [(17,2),(17,16)])
floor_map_data = generate_wall(floor_map_data,wall = [(14,4),(17,4)])
floor_map_data = generate_wall(floor_map_data,wall = [(12,4),(12,7)])
floor_map_data = generate_wall(floor_map_data,wall = [(12,7),(17,7)])
```

```
plt.imshow(floor_map_data)
```

```
<matplotlib.image.AxesImage at 0x2ae4576a208>
```




```
def generate_floor_map_point(floor_map,floor_id_list):
    result = []
    id_coordinate = 1
    for floor_id in floor_id_list:
        for y, y_values in enumerate(floor_map):
            for x, value in enumerate(y_values):
                passable = True if value == 0 else False
                result.append([id_coordinate,floor_id,x,y,passable])
                id_coordinate+=1
    return result
```

```
floor_map_points_data = generate_floor_map_point(floor_map_data,[1,2,3,4,5])
floor_map_points_columns = ['coordinatePoint','idfloorMap','x','y','ifpassable']
```

```
len(floor_map_points_data)
```

```
2100
```

```
20*21*5
```

```
2100
```

```
floor_map_points = pd.DataFrame(data = floor_map_points_data,columns=floor_map_points_columns)
```

```
# Insert data into database
pd.io.sql.to_sql(floor_map_points,'floorMapPoints',connection,schema='mydb',if_exists='append',index
=False)
```

```
%%sql
select * from floorMapPoints
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
2100 rows affected.
```

coordinatePoint	idfloorMap	x	y	ifpassable
1	1	0	0	1
2	1	1	0	1
3	1	2	0	1
4	1	3	0	1
5	1	4	0	1
6	1	5	0	1
7	1	6	0	1
8	1	7	0	1
9	1	8	0	1
10	1	9	0	1
11	1	10	0	1
12	1	11	0	1
13	1	12	0	1
14	1	13	0	1
15	1	14	0	1
16	1	15	0	1
17	1	16	0	1
18	1	17	0	1
19	1	18	0	1
20	1	19	0	1
21	1	20	0	1
22	1	0	1	1
23	1	1	1	1
24	1	2	1	1
25	1	3	1	1
26	1	4	1	1
27	1	5	1	1
28	1	6	1	1
29	1	7	1	1
30	1	8	1	1
31	1	9	1	1
32	1	10	1	1
33	1	11	1	1
34	1	12	1	1
35	1	13	1	1
36	1	14	1	1
37	1	15	1	1
38	1	16	1	1
39	1	17	1	1

40	1	18	1	1
41	1	19	1	1
42	1	20	1	1
43	1	0	2	1
44	1	1	2	1
45	1	2	2	0
46	1	3	2	0
47	1	4	2	0
48	1	5	2	0
49	1	6	2	0
50	1	7	2	0
51	1	8	2	0
52	1	9	2	0
53	1	10	2	1
54	1	11	2	0
55	1	12	2	0
56	1	13	2	0
57	1	14	2	0
58	1	15	2	0
59	1	16	2	0
60	1	17	2	0
61	1	18	2	1
62	1	19	2	1
63	1	20	2	1
64	1	0	3	1
65	1	1	3	1
66	1	2	3	0
67	1	3	3	1
68	1	4	3	1
69	1	5	3	1
70	1	6	3	1
71	1	7	3	1
72	1	8	3	1
73	1	9	3	0
74	1	10	3	1
75	1	11	3	1
76	1	12	3	1
77	1	13	3	1
78	1	14	3	1
79	1	15	3	1

80	1	16	3	1
81	1	17	3	0
82	1	18	3	1
83	1	19	3	1
84	1	20	3	1
85	1	0	4	1
86	1	1	4	1
87	1	2	4	0
88	1	3	4	1
89	1	4	4	1
90	1	5	4	1
91	1	6	4	1
92	1	7	4	1
93	1	8	4	1
94	1	9	4	0
95	1	10	4	1
96	1	11	4	1
97	1	12	4	0
98	1	13	4	1
99	1	14	4	0
100	1	15	4	0
101	1	16	4	0
102	1	17	4	0
103	1	18	4	1
104	1	19	4	1
105	1	20	4	1
106	1	0	5	1
107	1	1	5	1
108	1	2	5	0
109	1	3	5	1
110	1	4	5	1
111	1	5	5	1
112	1	6	5	1
113	1	7	5	1
114	1	8	5	1
115	1	9	5	0
116	1	10	5	1
117	1	11	5	1
118	1	12	5	0
119	1	13	5	1

120	1	14	5	1
121	1	15	5	1
122	1	16	5	1
123	1	17	5	0
124	1	18	5	1
125	1	19	5	1
126	1	20	5	1
127	1	0	6	1
128	1	1	6	1
129	1	2	6	0
130	1	3	6	1
131	1	4	6	1
132	1	5	6	1
133	1	6	6	1
134	1	7	6	1
135	1	8	6	1
136	1	9	6	0
137	1	10	6	1
138	1	11	6	1
139	1	12	6	0
140	1	13	6	1
141	1	14	6	1
142	1	15	6	1
143	1	16	6	1
144	1	17	6	0
145	1	18	6	1
146	1	19	6	1
147	1	20	6	1
148	1	0	7	1
149	1	1	7	1
150	1	2	7	0
151	1	3	7	1
152	1	4	7	1
153	1	5	7	1
154	1	6	7	1
155	1	7	7	1
156	1	8	7	1
157	1	9	7	0
158	1	10	7	1
159	1	11	7	1

160	1	12	7	0
161	1	13	7	0
162	1	14	7	0
163	1	15	7	0
164	1	16	7	0
165	1	17	7	0
166	1	18	7	1
167	1	19	7	1
168	1	20	7	1
169	1	0	8	1
170	1	1	8	1
171	1	2	8	0
172	1	3	8	1
173	1	4	8	1
174	1	5	8	1
175	1	6	8	1
176	1	7	8	1
177	1	8	8	1
178	1	9	8	0
179	1	10	8	1
180	1	11	8	1
181	1	12	8	1
182	1	13	8	1
183	1	14	8	1
184	1	15	8	1
185	1	16	8	1
186	1	17	8	0
187	1	18	8	1
188	1	19	8	1
189	1	20	8	1
190	1	0	9	1
191	1	1	9	1
192	1	2	9	0
193	1	3	9	1
194	1	4	9	1
195	1	5	9	1
196	1	6	9	1
197	1	7	9	1
198	1	8	9	1
199	1	9	9	1

200	1	10	9	1
201	1	11	9	1
202	1	12	9	1
203	1	13	9	1
204	1	14	9	1
205	1	15	9	1
206	1	16	9	1
207	1	17	9	0
208	1	18	9	1
209	1	19	9	1
210	1	20	9	1
211	1	0	10	1
212	1	1	10	1
213	1	2	10	0
214	1	3	10	0
215	1	4	10	0
216	1	5	10	0
217	1	6	10	0
218	1	7	10	0
219	1	8	10	0
220	1	9	10	0
221	1	10	10	1
222	1	11	10	1
223	1	12	10	1
224	1	13	10	1
225	1	14	10	1
226	1	15	10	1
227	1	16	10	1
228	1	17	10	0
229	1	18	10	1
230	1	19	10	1
231	1	20	10	1
232	1	0	11	1
233	1	1	11	1
234	1	2	11	0
235	1	3	11	1
236	1	4	11	1
237	1	5	11	1
238	1	6	11	1
239	1	7	11	1

240	1	8	11	1
241	1	9	11	1
242	1	10	11	1
243	1	11	11	1
244	1	12	11	1
245	1	13	11	1
246	1	14	11	1
247	1	15	11	1
248	1	16	11	1
249	1	17	11	0
250	1	18	11	1
251	1	19	11	1
252	1	20	11	1
253	1	0	12	1
254	1	1	12	1
255	1	2	12	0
256	1	3	12	1
257	1	4	12	1
258	1	5	12	1
259	1	6	12	1
260	1	7	12	1
261	1	8	12	1
262	1	9	12	1
263	1	10	12	1
264	1	11	12	1
265	1	12	12	1
266	1	13	12	1
267	1	14	12	1
268	1	15	12	1
269	1	16	12	1
270	1	17	12	0
271	1	18	12	1
272	1	19	12	1
273	1	20	12	1
274	1	0	13	1
275	1	1	13	1
276	1	2	13	0
277	1	3	13	1
278	1	4	13	1
279	1	5	13	1

280	1	6	13	1
281	1	7	13	1
282	1	8	13	1
283	1	9	13	0
284	1	10	13	1
285	1	11	13	1
286	1	12	13	1
287	1	13	13	1
288	1	14	13	1
289	1	15	13	1
290	1	16	13	1
291	1	17	13	0
292	1	18	13	1
293	1	19	13	1
294	1	20	13	1
295	1	0	14	1
296	1	1	14	1
297	1	2	14	0
298	1	3	14	1
299	1	4	14	1
300	1	5	14	1
301	1	6	14	1
302	1	7	14	1
303	1	8	14	1
304	1	9	14	0
305	1	10	14	1
306	1	11	14	1
307	1	12	14	1
308	1	13	14	1
309	1	14	14	1
310	1	15	14	1
311	1	16	14	1
312	1	17	14	0
313	1	18	14	1
314	1	19	14	1
315	1	20	14	1
316	1	0	15	1
317	1	1	15	1
318	1	2	15	0
319	1	3	15	1

320	1	4	15	1
321	1	5	15	1
322	1	6	15	1
323	1	7	15	1
324	1	8	15	1
325	1	9	15	0
326	1	10	15	1
327	1	11	15	1
328	1	12	15	1
329	1	13	15	1
330	1	14	15	1
331	1	15	15	1
332	1	16	15	1
333	1	17	15	0
334	1	18	15	1
335	1	19	15	1
336	1	20	15	1
337	1	0	16	1
338	1	1	16	1
339	1	2	16	0
340	1	3	16	1
341	1	4	16	1
342	1	5	16	1
343	1	6	16	1
344	1	7	16	1
345	1	8	16	1
346	1	9	16	0
347	1	10	16	0
348	1	11	16	0
349	1	12	16	0
350	1	13	16	0
351	1	14	16	0
352	1	15	16	0
353	1	16	16	0
354	1	17	16	0
355	1	18	16	1
356	1	19	16	1
357	1	20	16	1
358	1	0	17	1
359	1	1	17	1

360	1	2	17	0
361	1	3	17	1
362	1	4	17	1
363	1	5	17	1
364	1	6	17	1
365	1	7	17	1
366	1	8	17	1
367	1	9	17	1
368	1	10	17	1
369	1	11	17	1
370	1	12	17	1
371	1	13	17	1
372	1	14	17	1
373	1	15	17	1
374	1	16	17	1
375	1	17	17	1
376	1	18	17	1
377	1	19	17	1
378	1	20	17	1
379	1	0	18	1
380	1	1	18	1
381	1	2	18	0
382	1	3	18	0
383	1	4	18	0
384	1	5	18	0
385	1	6	18	0
386	1	7	18	0
387	1	8	18	0
388	1	9	18	0
389	1	10	18	1
390	1	11	18	1
391	1	12	18	1
392	1	13	18	1
393	1	14	18	1
394	1	15	18	1
395	1	16	18	1
396	1	17	18	1
397	1	18	18	1
398	1	19	18	1
399	1	20	18	1

400	1	0	19	1
401	1	1	19	1
402	1	2	19	1
403	1	3	19	1
404	1	4	19	1
405	1	5	19	1
406	1	6	19	1
407	1	7	19	1
408	1	8	19	1
409	1	9	19	1
410	1	10	19	1
411	1	11	19	1
412	1	12	19	1
413	1	13	19	1
414	1	14	19	1
415	1	15	19	1
416	1	16	19	1
417	1	17	19	1
418	1	18	19	1
419	1	19	19	1
420	1	20	19	1
421	2	0	0	1
422	2	1	0	1
423	2	2	0	1
424	2	3	0	1
425	2	4	0	1
426	2	5	0	1
427	2	6	0	1
428	2	7	0	1
429	2	8	0	1
430	2	9	0	1
431	2	10	0	1
432	2	11	0	1
433	2	12	0	1
434	2	13	0	1
435	2	14	0	1
436	2	15	0	1
437	2	16	0	1
438	2	17	0	1
439	2	18	0	1

440	2	19	0	1
441	2	20	0	1
442	2	0	1	1
443	2	1	1	1
444	2	2	1	1
445	2	3	1	1
446	2	4	1	1
447	2	5	1	1
448	2	6	1	1
449	2	7	1	1
450	2	8	1	1
451	2	9	1	1
452	2	10	1	1
453	2	11	1	1
454	2	12	1	1
455	2	13	1	1
456	2	14	1	1
457	2	15	1	1
458	2	16	1	1
459	2	17	1	1
460	2	18	1	1
461	2	19	1	1
462	2	20	1	1
463	2	0	2	1
464	2	1	2	1
465	2	2	2	0
466	2	3	2	0
467	2	4	2	0
468	2	5	2	0
469	2	6	2	0
470	2	7	2	0
471	2	8	2	0
472	2	9	2	0
473	2	10	2	1
474	2	11	2	0
475	2	12	2	0
476	2	13	2	0
477	2	14	2	0
478	2	15	2	0
479	2	16	2	0

480	2	17	2	0
481	2	18	2	1
482	2	19	2	1
483	2	20	2	1
484	2	0	3	1
485	2	1	3	1
486	2	2	3	0
487	2	3	3	1
488	2	4	3	1
489	2	5	3	1
490	2	6	3	1
491	2	7	3	1
492	2	8	3	1
493	2	9	3	0
494	2	10	3	1
495	2	11	3	1
496	2	12	3	1
497	2	13	3	1
498	2	14	3	1
499	2	15	3	1
500	2	16	3	1
501	2	17	3	0
502	2	18	3	1
503	2	19	3	1
504	2	20	3	1
505	2	0	4	1
506	2	1	4	1
507	2	2	4	0
508	2	3	4	1
509	2	4	4	1
510	2	5	4	1
511	2	6	4	1
512	2	7	4	1
513	2	8	4	1
514	2	9	4	0
515	2	10	4	1
516	2	11	4	1
517	2	12	4	0
518	2	13	4	1
519	2	14	4	0

520	2	15	4	0
521	2	16	4	0
522	2	17	4	0
523	2	18	4	1
524	2	19	4	1
525	2	20	4	1
526	2	0	5	1
527	2	1	5	1
528	2	2	5	0
529	2	3	5	1
530	2	4	5	1
531	2	5	5	1
532	2	6	5	1
533	2	7	5	1
534	2	8	5	1
535	2	9	5	0
536	2	10	5	1
537	2	11	5	1
538	2	12	5	0
539	2	13	5	1
540	2	14	5	1
541	2	15	5	1
542	2	16	5	1
543	2	17	5	0
544	2	18	5	1
545	2	19	5	1
546	2	20	5	1
547	2	0	6	1
548	2	1	6	1
549	2	2	6	0
550	2	3	6	1
551	2	4	6	1
552	2	5	6	1
553	2	6	6	1
554	2	7	6	1
555	2	8	6	1
556	2	9	6	0
557	2	10	6	1
558	2	11	6	1
559	2	12	6	0

560	2	13	6	1
561	2	14	6	1
562	2	15	6	1
563	2	16	6	1
564	2	17	6	0
565	2	18	6	1
566	2	19	6	1
567	2	20	6	1
568	2	0	7	1
569	2	1	7	1
570	2	2	7	0
571	2	3	7	1
572	2	4	7	1
573	2	5	7	1
574	2	6	7	1
575	2	7	7	1
576	2	8	7	1
577	2	9	7	0
578	2	10	7	1
579	2	11	7	1
580	2	12	7	0
581	2	13	7	0
582	2	14	7	0
583	2	15	7	0
584	2	16	7	0
585	2	17	7	0
586	2	18	7	1
587	2	19	7	1
588	2	20	7	1
589	2	0	8	1
590	2	1	8	1
591	2	2	8	0
592	2	3	8	1
593	2	4	8	1
594	2	5	8	1
595	2	6	8	1
596	2	7	8	1
597	2	8	8	1
598	2	9	8	0
599	2	10	8	1

600	2	11	8	1
601	2	12	8	1
602	2	13	8	1
603	2	14	8	1
604	2	15	8	1
605	2	16	8	1
606	2	17	8	0
607	2	18	8	1
608	2	19	8	1
609	2	20	8	1
610	2	0	9	1
611	2	1	9	1
612	2	2	9	0
613	2	3	9	1
614	2	4	9	1
615	2	5	9	1
616	2	6	9	1
617	2	7	9	1
618	2	8	9	1
619	2	9	9	1
620	2	10	9	1
621	2	11	9	1
622	2	12	9	1
623	2	13	9	1
624	2	14	9	1
625	2	15	9	1
626	2	16	9	1
627	2	17	9	0
628	2	18	9	1
629	2	19	9	1
630	2	20	9	1
631	2	0	10	1
632	2	1	10	1
633	2	2	10	0
634	2	3	10	0
635	2	4	10	0
636	2	5	10	0
637	2	6	10	0
638	2	7	10	0
639	2	8	10	0

640	2	9	10	0
641	2	10	10	1
642	2	11	10	1
643	2	12	10	1
644	2	13	10	1
645	2	14	10	1
646	2	15	10	1
647	2	16	10	1
648	2	17	10	0
649	2	18	10	1
650	2	19	10	1
651	2	20	10	1
652	2	0	11	1
653	2	1	11	1
654	2	2	11	0
655	2	3	11	1
656	2	4	11	1
657	2	5	11	1
658	2	6	11	1
659	2	7	11	1
660	2	8	11	1
661	2	9	11	1
662	2	10	11	1
663	2	11	11	1
664	2	12	11	1
665	2	13	11	1
666	2	14	11	1
667	2	15	11	1
668	2	16	11	1
669	2	17	11	0
670	2	18	11	1
671	2	19	11	1
672	2	20	11	1
673	2	0	12	1
674	2	1	12	1
675	2	2	12	0
676	2	3	12	1
677	2	4	12	1
678	2	5	12	1
679	2	6	12	1

680	2	7	12	1
681	2	8	12	1
682	2	9	12	1
683	2	10	12	1
684	2	11	12	1
685	2	12	12	1
686	2	13	12	1
687	2	14	12	1
688	2	15	12	1
689	2	16	12	1
690	2	17	12	0
691	2	18	12	1
692	2	19	12	1
693	2	20	12	1
694	2	0	13	1
695	2	1	13	1
696	2	2	13	0
697	2	3	13	1
698	2	4	13	1
699	2	5	13	1
700	2	6	13	1
701	2	7	13	1
702	2	8	13	1
703	2	9	13	0
704	2	10	13	1
705	2	11	13	1
706	2	12	13	1
707	2	13	13	1
708	2	14	13	1
709	2	15	13	1
710	2	16	13	1
711	2	17	13	0
712	2	18	13	1
713	2	19	13	1
714	2	20	13	1
715	2	0	14	1
716	2	1	14	1
717	2	2	14	0
718	2	3	14	1
719	2	4	14	1

720	2	5	14	1
721	2	6	14	1
722	2	7	14	1
723	2	8	14	1
724	2	9	14	0
725	2	10	14	1
726	2	11	14	1
727	2	12	14	1
728	2	13	14	1
729	2	14	14	1
730	2	15	14	1
731	2	16	14	1
732	2	17	14	0
733	2	18	14	1
734	2	19	14	1
735	2	20	14	1
736	2	0	15	1
737	2	1	15	1
738	2	2	15	0
739	2	3	15	1
740	2	4	15	1
741	2	5	15	1
742	2	6	15	1
743	2	7	15	1
744	2	8	15	1
745	2	9	15	0
746	2	10	15	1
747	2	11	15	1
748	2	12	15	1
749	2	13	15	1
750	2	14	15	1
751	2	15	15	1
752	2	16	15	1
753	2	17	15	0
754	2	18	15	1
755	2	19	15	1
756	2	20	15	1
757	2	0	16	1
758	2	1	16	1
759	2	2	16	0

760	2	3	16	1
761	2	4	16	1
762	2	5	16	1
763	2	6	16	1
764	2	7	16	1
765	2	8	16	1
766	2	9	16	0
767	2	10	16	0
768	2	11	16	0
769	2	12	16	0
770	2	13	16	0
771	2	14	16	0
772	2	15	16	0
773	2	16	16	0
774	2	17	16	0
775	2	18	16	1
776	2	19	16	1
777	2	20	16	1
778	2	0	17	1
779	2	1	17	1
780	2	2	17	0
781	2	3	17	1
782	2	4	17	1
783	2	5	17	1
784	2	6	17	1
785	2	7	17	1
786	2	8	17	1
787	2	9	17	1
788	2	10	17	1
789	2	11	17	1
790	2	12	17	1
791	2	13	17	1
792	2	14	17	1
793	2	15	17	1
794	2	16	17	1
795	2	17	17	1
796	2	18	17	1
797	2	19	17	1
798	2	20	17	1
799	2	0	18	1

800	2	1	18	1
801	2	2	18	0
802	2	3	18	0
803	2	4	18	0
804	2	5	18	0
805	2	6	18	0
806	2	7	18	0
807	2	8	18	0
808	2	9	18	0
809	2	10	18	1
810	2	11	18	1
811	2	12	18	1
812	2	13	18	1
813	2	14	18	1
814	2	15	18	1
815	2	16	18	1
816	2	17	18	1
817	2	18	18	1
818	2	19	18	1
819	2	20	18	1
820	2	0	19	1
821	2	1	19	1
822	2	2	19	1
823	2	3	19	1
824	2	4	19	1
825	2	5	19	1
826	2	6	19	1
827	2	7	19	1
828	2	8	19	1
829	2	9	19	1
830	2	10	19	1
831	2	11	19	1
832	2	12	19	1
833	2	13	19	1
834	2	14	19	1
835	2	15	19	1
836	2	16	19	1
837	2	17	19	1
838	2	18	19	1
839	2	19	19	1

840	2	20	19	1
841	3	0	0	1
842	3	1	0	1
843	3	2	0	1
844	3	3	0	1
845	3	4	0	1
846	3	5	0	1
847	3	6	0	1
848	3	7	0	1
849	3	8	0	1
850	3	9	0	1
851	3	10	0	1
852	3	11	0	1
853	3	12	0	1
854	3	13	0	1
855	3	14	0	1
856	3	15	0	1
857	3	16	0	1
858	3	17	0	1
859	3	18	0	1
860	3	19	0	1
861	3	20	0	1
862	3	0	1	1
863	3	1	1	1
864	3	2	1	1
865	3	3	1	1
866	3	4	1	1
867	3	5	1	1
868	3	6	1	1
869	3	7	1	1
870	3	8	1	1
871	3	9	1	1
872	3	10	1	1
873	3	11	1	1
874	3	12	1	1
875	3	13	1	1
876	3	14	1	1
877	3	15	1	1
878	3	16	1	1
879	3	17	1	1

880	3	18	1	1
881	3	19	1	1
882	3	20	1	1
883	3	0	2	1
884	3	1	2	1
885	3	2	2	0
886	3	3	2	0
887	3	4	2	0
888	3	5	2	0
889	3	6	2	0
890	3	7	2	0
891	3	8	2	0
892	3	9	2	0
893	3	10	2	1
894	3	11	2	0
895	3	12	2	0
896	3	13	2	0
897	3	14	2	0
898	3	15	2	0
899	3	16	2	0
900	3	17	2	0
901	3	18	2	1
902	3	19	2	1
903	3	20	2	1
904	3	0	3	1
905	3	1	3	1
906	3	2	3	0
907	3	3	3	1
908	3	4	3	1
909	3	5	3	1
910	3	6	3	1
911	3	7	3	1
912	3	8	3	1
913	3	9	3	0
914	3	10	3	1
915	3	11	3	1
916	3	12	3	1
917	3	13	3	1
918	3	14	3	1
919	3	15	3	1

920	3	16	3	1
921	3	17	3	0
922	3	18	3	1
923	3	19	3	1
924	3	20	3	1
925	3	0	4	1
926	3	1	4	1
927	3	2	4	0
928	3	3	4	1
929	3	4	4	1
930	3	5	4	1
931	3	6	4	1
932	3	7	4	1
933	3	8	4	1
934	3	9	4	0
935	3	10	4	1
936	3	11	4	1
937	3	12	4	0
938	3	13	4	1
939	3	14	4	0
940	3	15	4	0
941	3	16	4	0
942	3	17	4	0
943	3	18	4	1
944	3	19	4	1
945	3	20	4	1
946	3	0	5	1
947	3	1	5	1
948	3	2	5	0
949	3	3	5	1
950	3	4	5	1
951	3	5	5	1
952	3	6	5	1
953	3	7	5	1
954	3	8	5	1
955	3	9	5	0
956	3	10	5	1
957	3	11	5	1
958	3	12	5	0
959	3	13	5	1

960	3	14	5	1
961	3	15	5	1
962	3	16	5	1
963	3	17	5	0
964	3	18	5	1
965	3	19	5	1
966	3	20	5	1
967	3	0	6	1
968	3	1	6	1
969	3	2	6	0
970	3	3	6	1
971	3	4	6	1
972	3	5	6	1
973	3	6	6	1
974	3	7	6	1
975	3	8	6	1
976	3	9	6	0
977	3	10	6	1
978	3	11	6	1
979	3	12	6	0
980	3	13	6	1
981	3	14	6	1
982	3	15	6	1
983	3	16	6	1
984	3	17	6	0
985	3	18	6	1
986	3	19	6	1
987	3	20	6	1
988	3	0	7	1
989	3	1	7	1
990	3	2	7	0
991	3	3	7	1
992	3	4	7	1
993	3	5	7	1
994	3	6	7	1
995	3	7	7	1
996	3	8	7	1
997	3	9	7	0
998	3	10	7	1
999	3	11	7	1

1000	3	12	7	0
1001	3	13	7	0
1002	3	14	7	0
1003	3	15	7	0
1004	3	16	7	0
1005	3	17	7	0
1006	3	18	7	1
1007	3	19	7	1
1008	3	20	7	1
1009	3	0	8	1
1010	3	1	8	1
1011	3	2	8	0
1012	3	3	8	1
1013	3	4	8	1
1014	3	5	8	1
1015	3	6	8	1
1016	3	7	8	1
1017	3	8	8	1
1018	3	9	8	0
1019	3	10	8	1
1020	3	11	8	1
1021	3	12	8	1
1022	3	13	8	1
1023	3	14	8	1
1024	3	15	8	1
1025	3	16	8	1
1026	3	17	8	0
1027	3	18	8	1
1028	3	19	8	1
1029	3	20	8	1
1030	3	0	9	1
1031	3	1	9	1
1032	3	2	9	0
1033	3	3	9	1
1034	3	4	9	1
1035	3	5	9	1
1036	3	6	9	1
1037	3	7	9	1
1038	3	8	9	1
1039	3	9	9	1

1040	3	10	9	1
1041	3	11	9	1
1042	3	12	9	1
1043	3	13	9	1
1044	3	14	9	1
1045	3	15	9	1
1046	3	16	9	1
1047	3	17	9	0
1048	3	18	9	1
1049	3	19	9	1
1050	3	20	9	1
1051	3	0	10	1
1052	3	1	10	1
1053	3	2	10	0
1054	3	3	10	0
1055	3	4	10	0
1056	3	5	10	0
1057	3	6	10	0
1058	3	7	10	0
1059	3	8	10	0
1060	3	9	10	0
1061	3	10	10	1
1062	3	11	10	1
1063	3	12	10	1
1064	3	13	10	1
1065	3	14	10	1
1066	3	15	10	1
1067	3	16	10	1
1068	3	17	10	0
1069	3	18	10	1
1070	3	19	10	1
1071	3	20	10	1
1072	3	0	11	1
1073	3	1	11	1
1074	3	2	11	0
1075	3	3	11	1
1076	3	4	11	1
1077	3	5	11	1
1078	3	6	11	1
1079	3	7	11	1

1080	3	8	11	1
1081	3	9	11	1
1082	3	10	11	1
1083	3	11	11	1
1084	3	12	11	1
1085	3	13	11	1
1086	3	14	11	1
1087	3	15	11	1
1088	3	16	11	1
1089	3	17	11	0
1090	3	18	11	1
1091	3	19	11	1
1092	3	20	11	1
1093	3	0	12	1
1094	3	1	12	1
1095	3	2	12	0
1096	3	3	12	1
1097	3	4	12	1
1098	3	5	12	1
1099	3	6	12	1
1100	3	7	12	1
1101	3	8	12	1
1102	3	9	12	1
1103	3	10	12	1
1104	3	11	12	1
1105	3	12	12	1
1106	3	13	12	1
1107	3	14	12	1
1108	3	15	12	1
1109	3	16	12	1
1110	3	17	12	0
1111	3	18	12	1
1112	3	19	12	1
1113	3	20	12	1
1114	3	0	13	1
1115	3	1	13	1
1116	3	2	13	0
1117	3	3	13	1
1118	3	4	13	1
1119	3	5	13	1

1120	3	6	13	1
1121	3	7	13	1
1122	3	8	13	1
1123	3	9	13	0
1124	3	10	13	1
1125	3	11	13	1
1126	3	12	13	1
1127	3	13	13	1
1128	3	14	13	1
1129	3	15	13	1
1130	3	16	13	1
1131	3	17	13	0
1132	3	18	13	1
1133	3	19	13	1
1134	3	20	13	1
1135	3	0	14	1
1136	3	1	14	1
1137	3	2	14	0
1138	3	3	14	1
1139	3	4	14	1
1140	3	5	14	1
1141	3	6	14	1
1142	3	7	14	1
1143	3	8	14	1
1144	3	9	14	0
1145	3	10	14	1
1146	3	11	14	1
1147	3	12	14	1
1148	3	13	14	1
1149	3	14	14	1
1150	3	15	14	1
1151	3	16	14	1
1152	3	17	14	0
1153	3	18	14	1
1154	3	19	14	1
1155	3	20	14	1
1156	3	0	15	1
1157	3	1	15	1
1158	3	2	15	0
1159	3	3	15	1

1160	3	4	15	1
1161	3	5	15	1
1162	3	6	15	1
1163	3	7	15	1
1164	3	8	15	1
1165	3	9	15	0
1166	3	10	15	1
1167	3	11	15	1
1168	3	12	15	1
1169	3	13	15	1
1170	3	14	15	1
1171	3	15	15	1
1172	3	16	15	1
1173	3	17	15	0
1174	3	18	15	1
1175	3	19	15	1
1176	3	20	15	1
1177	3	0	16	1
1178	3	1	16	1
1179	3	2	16	0
1180	3	3	16	1
1181	3	4	16	1
1182	3	5	16	1
1183	3	6	16	1
1184	3	7	16	1
1185	3	8	16	1
1186	3	9	16	0
1187	3	10	16	0
1188	3	11	16	0
1189	3	12	16	0
1190	3	13	16	0
1191	3	14	16	0
1192	3	15	16	0
1193	3	16	16	0
1194	3	17	16	0
1195	3	18	16	1
1196	3	19	16	1
1197	3	20	16	1
1198	3	0	17	1
1199	3	1	17	1

1200	3	2	17	0
1201	3	3	17	1
1202	3	4	17	1
1203	3	5	17	1
1204	3	6	17	1
1205	3	7	17	1
1206	3	8	17	1
1207	3	9	17	1
1208	3	10	17	1
1209	3	11	17	1
1210	3	12	17	1
1211	3	13	17	1
1212	3	14	17	1
1213	3	15	17	1
1214	3	16	17	1
1215	3	17	17	1
1216	3	18	17	1
1217	3	19	17	1
1218	3	20	17	1
1219	3	0	18	1
1220	3	1	18	1
1221	3	2	18	0
1222	3	3	18	0
1223	3	4	18	0
1224	3	5	18	0
1225	3	6	18	0
1226	3	7	18	0
1227	3	8	18	0
1228	3	9	18	0
1229	3	10	18	1
1230	3	11	18	1
1231	3	12	18	1
1232	3	13	18	1
1233	3	14	18	1
1234	3	15	18	1
1235	3	16	18	1
1236	3	17	18	1
1237	3	18	18	1
1238	3	19	18	1
1239	3	20	18	1

1240	3	0	19	1
1241	3	1	19	1
1242	3	2	19	1
1243	3	3	19	1
1244	3	4	19	1
1245	3	5	19	1
1246	3	6	19	1
1247	3	7	19	1
1248	3	8	19	1
1249	3	9	19	1
1250	3	10	19	1
1251	3	11	19	1
1252	3	12	19	1
1253	3	13	19	1
1254	3	14	19	1
1255	3	15	19	1
1256	3	16	19	1
1257	3	17	19	1
1258	3	18	19	1
1259	3	19	19	1
1260	3	20	19	1
1261	4	0	0	1
1262	4	1	0	1
1263	4	2	0	1
1264	4	3	0	1
1265	4	4	0	1
1266	4	5	0	1
1267	4	6	0	1
1268	4	7	0	1
1269	4	8	0	1
1270	4	9	0	1
1271	4	10	0	1
1272	4	11	0	1
1273	4	12	0	1
1274	4	13	0	1
1275	4	14	0	1
1276	4	15	0	1
1277	4	16	0	1
1278	4	17	0	1
1279	4	18	0	1

1280	4	19	0	1
1281	4	20	0	1
1282	4	0	1	1
1283	4	1	1	1
1284	4	2	1	1
1285	4	3	1	1
1286	4	4	1	1
1287	4	5	1	1
1288	4	6	1	1
1289	4	7	1	1
1290	4	8	1	1
1291	4	9	1	1
1292	4	10	1	1
1293	4	11	1	1
1294	4	12	1	1
1295	4	13	1	1
1296	4	14	1	1
1297	4	15	1	1
1298	4	16	1	1
1299	4	17	1	1
1300	4	18	1	1
1301	4	19	1	1
1302	4	20	1	1
1303	4	0	2	1
1304	4	1	2	1
1305	4	2	2	0
1306	4	3	2	0
1307	4	4	2	0
1308	4	5	2	0
1309	4	6	2	0
1310	4	7	2	0
1311	4	8	2	0
1312	4	9	2	0
1313	4	10	2	1
1314	4	11	2	0
1315	4	12	2	0
1316	4	13	2	0
1317	4	14	2	0
1318	4	15	2	0
1319	4	16	2	0

1320	4	17	2	0
1321	4	18	2	1
1322	4	19	2	1
1323	4	20	2	1
1324	4	0	3	1
1325	4	1	3	1
1326	4	2	3	0
1327	4	3	3	1
1328	4	4	3	1
1329	4	5	3	1
1330	4	6	3	1
1331	4	7	3	1
1332	4	8	3	1
1333	4	9	3	0
1334	4	10	3	1
1335	4	11	3	1
1336	4	12	3	1
1337	4	13	3	1
1338	4	14	3	1
1339	4	15	3	1
1340	4	16	3	1
1341	4	17	3	0
1342	4	18	3	1
1343	4	19	3	1
1344	4	20	3	1
1345	4	0	4	1
1346	4	1	4	1
1347	4	2	4	0
1348	4	3	4	1
1349	4	4	4	1
1350	4	5	4	1
1351	4	6	4	1
1352	4	7	4	1
1353	4	8	4	1
1354	4	9	4	0
1355	4	10	4	1
1356	4	11	4	1
1357	4	12	4	0
1358	4	13	4	1
1359	4	14	4	0

1360	4	15	4	0
1361	4	16	4	0
1362	4	17	4	0
1363	4	18	4	1
1364	4	19	4	1
1365	4	20	4	1
1366	4	0	5	1
1367	4	1	5	1
1368	4	2	5	0
1369	4	3	5	1
1370	4	4	5	1
1371	4	5	5	1
1372	4	6	5	1
1373	4	7	5	1
1374	4	8	5	1
1375	4	9	5	0
1376	4	10	5	1
1377	4	11	5	1
1378	4	12	5	0
1379	4	13	5	1
1380	4	14	5	1
1381	4	15	5	1
1382	4	16	5	1
1383	4	17	5	0
1384	4	18	5	1
1385	4	19	5	1
1386	4	20	5	1
1387	4	0	6	1
1388	4	1	6	1
1389	4	2	6	0
1390	4	3	6	1
1391	4	4	6	1
1392	4	5	6	1
1393	4	6	6	1
1394	4	7	6	1
1395	4	8	6	1
1396	4	9	6	0
1397	4	10	6	1
1398	4	11	6	1
1399	4	12	6	0

1400	4	13	6	1
1401	4	14	6	1
1402	4	15	6	1
1403	4	16	6	1
1404	4	17	6	0
1405	4	18	6	1
1406	4	19	6	1
1407	4	20	6	1
1408	4	0	7	1
1409	4	1	7	1
1410	4	2	7	0
1411	4	3	7	1
1412	4	4	7	1
1413	4	5	7	1
1414	4	6	7	1
1415	4	7	7	1
1416	4	8	7	1
1417	4	9	7	0
1418	4	10	7	1
1419	4	11	7	1
1420	4	12	7	0
1421	4	13	7	0
1422	4	14	7	0
1423	4	15	7	0
1424	4	16	7	0
1425	4	17	7	0
1426	4	18	7	1
1427	4	19	7	1
1428	4	20	7	1
1429	4	0	8	1
1430	4	1	8	1
1431	4	2	8	0
1432	4	3	8	1
1433	4	4	8	1
1434	4	5	8	1
1435	4	6	8	1
1436	4	7	8	1
1437	4	8	8	1
1438	4	9	8	0
1439	4	10	8	1

1440	4	11	8	1
1441	4	12	8	1
1442	4	13	8	1
1443	4	14	8	1
1444	4	15	8	1
1445	4	16	8	1
1446	4	17	8	0
1447	4	18	8	1
1448	4	19	8	1
1449	4	20	8	1
1450	4	0	9	1
1451	4	1	9	1
1452	4	2	9	0
1453	4	3	9	1
1454	4	4	9	1
1455	4	5	9	1
1456	4	6	9	1
1457	4	7	9	1
1458	4	8	9	1
1459	4	9	9	1
1460	4	10	9	1
1461	4	11	9	1
1462	4	12	9	1
1463	4	13	9	1
1464	4	14	9	1
1465	4	15	9	1
1466	4	16	9	1
1467	4	17	9	0
1468	4	18	9	1
1469	4	19	9	1
1470	4	20	9	1
1471	4	0	10	1
1472	4	1	10	1
1473	4	2	10	0
1474	4	3	10	0
1475	4	4	10	0
1476	4	5	10	0
1477	4	6	10	0
1478	4	7	10	0
1479	4	8	10	0

1480	4	9	10	0
1481	4	10	10	1
1482	4	11	10	1
1483	4	12	10	1
1484	4	13	10	1
1485	4	14	10	1
1486	4	15	10	1
1487	4	16	10	1
1488	4	17	10	0
1489	4	18	10	1
1490	4	19	10	1
1491	4	20	10	1
1492	4	0	11	1
1493	4	1	11	1
1494	4	2	11	0
1495	4	3	11	1
1496	4	4	11	1
1497	4	5	11	1
1498	4	6	11	1
1499	4	7	11	1
1500	4	8	11	1
1501	4	9	11	1
1502	4	10	11	1
1503	4	11	11	1
1504	4	12	11	1
1505	4	13	11	1
1506	4	14	11	1
1507	4	15	11	1
1508	4	16	11	1
1509	4	17	11	0
1510	4	18	11	1
1511	4	19	11	1
1512	4	20	11	1
1513	4	0	12	1
1514	4	1	12	1
1515	4	2	12	0
1516	4	3	12	1
1517	4	4	12	1
1518	4	5	12	1
1519	4	6	12	1

1520	4	7	12	1
1521	4	8	12	1
1522	4	9	12	1
1523	4	10	12	1
1524	4	11	12	1
1525	4	12	12	1
1526	4	13	12	1
1527	4	14	12	1
1528	4	15	12	1
1529	4	16	12	1
1530	4	17	12	0
1531	4	18	12	1
1532	4	19	12	1
1533	4	20	12	1
1534	4	0	13	1
1535	4	1	13	1
1536	4	2	13	0
1537	4	3	13	1
1538	4	4	13	1
1539	4	5	13	1
1540	4	6	13	1
1541	4	7	13	1
1542	4	8	13	1
1543	4	9	13	0
1544	4	10	13	1
1545	4	11	13	1
1546	4	12	13	1
1547	4	13	13	1
1548	4	14	13	1
1549	4	15	13	1
1550	4	16	13	1
1551	4	17	13	0
1552	4	18	13	1
1553	4	19	13	1
1554	4	20	13	1
1555	4	0	14	1
1556	4	1	14	1
1557	4	2	14	0
1558	4	3	14	1
1559	4	4	14	1

1560	4	5	14	1
1561	4	6	14	1
1562	4	7	14	1
1563	4	8	14	1
1564	4	9	14	0
1565	4	10	14	1
1566	4	11	14	1
1567	4	12	14	1
1568	4	13	14	1
1569	4	14	14	1
1570	4	15	14	1
1571	4	16	14	1
1572	4	17	14	0
1573	4	18	14	1
1574	4	19	14	1
1575	4	20	14	1
1576	4	0	15	1
1577	4	1	15	1
1578	4	2	15	0
1579	4	3	15	1
1580	4	4	15	1
1581	4	5	15	1
1582	4	6	15	1
1583	4	7	15	1
1584	4	8	15	1
1585	4	9	15	0
1586	4	10	15	1
1587	4	11	15	1
1588	4	12	15	1
1589	4	13	15	1
1590	4	14	15	1
1591	4	15	15	1
1592	4	16	15	1
1593	4	17	15	0
1594	4	18	15	1
1595	4	19	15	1
1596	4	20	15	1
1597	4	0	16	1
1598	4	1	16	1
1599	4	2	16	0

1600	4	3	16	1
1601	4	4	16	1
1602	4	5	16	1
1603	4	6	16	1
1604	4	7	16	1
1605	4	8	16	1
1606	4	9	16	0
1607	4	10	16	0
1608	4	11	16	0
1609	4	12	16	0
1610	4	13	16	0
1611	4	14	16	0
1612	4	15	16	0
1613	4	16	16	0
1614	4	17	16	0
1615	4	18	16	1
1616	4	19	16	1
1617	4	20	16	1
1618	4	0	17	1
1619	4	1	17	1
1620	4	2	17	0
1621	4	3	17	1
1622	4	4	17	1
1623	4	5	17	1
1624	4	6	17	1
1625	4	7	17	1
1626	4	8	17	1
1627	4	9	17	1
1628	4	10	17	1
1629	4	11	17	1
1630	4	12	17	1
1631	4	13	17	1
1632	4	14	17	1
1633	4	15	17	1
1634	4	16	17	1
1635	4	17	17	1
1636	4	18	17	1
1637	4	19	17	1
1638	4	20	17	1
1639	4	0	18	1

1640	4	1	18	1
1641	4	2	18	0
1642	4	3	18	0
1643	4	4	18	0
1644	4	5	18	0
1645	4	6	18	0
1646	4	7	18	0
1647	4	8	18	0
1648	4	9	18	0
1649	4	10	18	1
1650	4	11	18	1
1651	4	12	18	1
1652	4	13	18	1
1653	4	14	18	1
1654	4	15	18	1
1655	4	16	18	1
1656	4	17	18	1
1657	4	18	18	1
1658	4	19	18	1
1659	4	20	18	1
1660	4	0	19	1
1661	4	1	19	1
1662	4	2	19	1
1663	4	3	19	1
1664	4	4	19	1
1665	4	5	19	1
1666	4	6	19	1
1667	4	7	19	1
1668	4	8	19	1
1669	4	9	19	1
1670	4	10	19	1
1671	4	11	19	1
1672	4	12	19	1
1673	4	13	19	1
1674	4	14	19	1
1675	4	15	19	1
1676	4	16	19	1
1677	4	17	19	1
1678	4	18	19	1
1679	4	19	19	1

1680	4	20	19	1
1681	5	0	0	1
1682	5	1	0	1
1683	5	2	0	1
1684	5	3	0	1
1685	5	4	0	1
1686	5	5	0	1
1687	5	6	0	1
1688	5	7	0	1
1689	5	8	0	1
1690	5	9	0	1
1691	5	10	0	1
1692	5	11	0	1
1693	5	12	0	1
1694	5	13	0	1
1695	5	14	0	1
1696	5	15	0	1
1697	5	16	0	1
1698	5	17	0	1
1699	5	18	0	1
1700	5	19	0	1
1701	5	20	0	1
1702	5	0	1	1
1703	5	1	1	1
1704	5	2	1	1
1705	5	3	1	1
1706	5	4	1	1
1707	5	5	1	1
1708	5	6	1	1
1709	5	7	1	1
1710	5	8	1	1
1711	5	9	1	1
1712	5	10	1	1
1713	5	11	1	1
1714	5	12	1	1
1715	5	13	1	1
1716	5	14	1	1
1717	5	15	1	1
1718	5	16	1	1
1719	5	17	1	1

1720	5	18	1	1
1721	5	19	1	1
1722	5	20	1	1
1723	5	0	2	1
1724	5	1	2	1
1725	5	2	2	0
1726	5	3	2	0
1727	5	4	2	0
1728	5	5	2	0
1729	5	6	2	0
1730	5	7	2	0
1731	5	8	2	0
1732	5	9	2	0
1733	5	10	2	1
1734	5	11	2	0
1735	5	12	2	0
1736	5	13	2	0
1737	5	14	2	0
1738	5	15	2	0
1739	5	16	2	0
1740	5	17	2	0
1741	5	18	2	1
1742	5	19	2	1
1743	5	20	2	1
1744	5	0	3	1
1745	5	1	3	1
1746	5	2	3	0
1747	5	3	3	1
1748	5	4	3	1
1749	5	5	3	1
1750	5	6	3	1
1751	5	7	3	1
1752	5	8	3	1
1753	5	9	3	0
1754	5	10	3	1
1755	5	11	3	1
1756	5	12	3	1
1757	5	13	3	1
1758	5	14	3	1
1759	5	15	3	1

1760	5	16	3	1
1761	5	17	3	0
1762	5	18	3	1
1763	5	19	3	1
1764	5	20	3	1
1765	5	0	4	1
1766	5	1	4	1
1767	5	2	4	0
1768	5	3	4	1
1769	5	4	4	1
1770	5	5	4	1
1771	5	6	4	1
1772	5	7	4	1
1773	5	8	4	1
1774	5	9	4	0
1775	5	10	4	1
1776	5	11	4	1
1777	5	12	4	0
1778	5	13	4	1
1779	5	14	4	0
1780	5	15	4	0
1781	5	16	4	0
1782	5	17	4	0
1783	5	18	4	1
1784	5	19	4	1
1785	5	20	4	1
1786	5	0	5	1
1787	5	1	5	1
1788	5	2	5	0
1789	5	3	5	1
1790	5	4	5	1
1791	5	5	5	1
1792	5	6	5	1
1793	5	7	5	1
1794	5	8	5	1
1795	5	9	5	0
1796	5	10	5	1
1797	5	11	5	1
1798	5	12	5	0
1799	5	13	5	1

1800	5	14	5	1
1801	5	15	5	1
1802	5	16	5	1
1803	5	17	5	0
1804	5	18	5	1
1805	5	19	5	1
1806	5	20	5	1
1807	5	0	6	1
1808	5	1	6	1
1809	5	2	6	0
1810	5	3	6	1
1811	5	4	6	1
1812	5	5	6	1
1813	5	6	6	1
1814	5	7	6	1
1815	5	8	6	1
1816	5	9	6	0
1817	5	10	6	1
1818	5	11	6	1
1819	5	12	6	0
1820	5	13	6	1
1821	5	14	6	1
1822	5	15	6	1
1823	5	16	6	1
1824	5	17	6	0
1825	5	18	6	1
1826	5	19	6	1
1827	5	20	6	1
1828	5	0	7	1
1829	5	1	7	1
1830	5	2	7	0
1831	5	3	7	1
1832	5	4	7	1
1833	5	5	7	1
1834	5	6	7	1
1835	5	7	7	1
1836	5	8	7	1
1837	5	9	7	0
1838	5	10	7	1
1839	5	11	7	1

1840	5	12	7	0
1841	5	13	7	0
1842	5	14	7	0
1843	5	15	7	0
1844	5	16	7	0
1845	5	17	7	0
1846	5	18	7	1
1847	5	19	7	1
1848	5	20	7	1
1849	5	0	8	1
1850	5	1	8	1
1851	5	2	8	0
1852	5	3	8	1
1853	5	4	8	1
1854	5	5	8	1
1855	5	6	8	1
1856	5	7	8	1
1857	5	8	8	1
1858	5	9	8	0
1859	5	10	8	1
1860	5	11	8	1
1861	5	12	8	1
1862	5	13	8	1
1863	5	14	8	1
1864	5	15	8	1
1865	5	16	8	1
1866	5	17	8	0
1867	5	18	8	1
1868	5	19	8	1
1869	5	20	8	1
1870	5	0	9	1
1871	5	1	9	1
1872	5	2	9	0
1873	5	3	9	1
1874	5	4	9	1
1875	5	5	9	1
1876	5	6	9	1
1877	5	7	9	1
1878	5	8	9	1
1879	5	9	9	1

1880	5	10	9	1
1881	5	11	9	1
1882	5	12	9	1
1883	5	13	9	1
1884	5	14	9	1
1885	5	15	9	1
1886	5	16	9	1
1887	5	17	9	0
1888	5	18	9	1
1889	5	19	9	1
1890	5	20	9	1
1891	5	0	10	1
1892	5	1	10	1
1893	5	2	10	0
1894	5	3	10	0
1895	5	4	10	0
1896	5	5	10	0
1897	5	6	10	0
1898	5	7	10	0
1899	5	8	10	0
1900	5	9	10	0
1901	5	10	10	1
1902	5	11	10	1
1903	5	12	10	1
1904	5	13	10	1
1905	5	14	10	1
1906	5	15	10	1
1907	5	16	10	1
1908	5	17	10	0
1909	5	18	10	1
1910	5	19	10	1
1911	5	20	10	1
1912	5	0	11	1
1913	5	1	11	1
1914	5	2	11	0
1915	5	3	11	1
1916	5	4	11	1
1917	5	5	11	1
1918	5	6	11	1
1919	5	7	11	1

1920	5	8	11	1
1921	5	9	11	1
1922	5	10	11	1
1923	5	11	11	1
1924	5	12	11	1
1925	5	13	11	1
1926	5	14	11	1
1927	5	15	11	1
1928	5	16	11	1
1929	5	17	11	0
1930	5	18	11	1
1931	5	19	11	1
1932	5	20	11	1
1933	5	0	12	1
1934	5	1	12	1
1935	5	2	12	0
1936	5	3	12	1
1937	5	4	12	1
1938	5	5	12	1
1939	5	6	12	1
1940	5	7	12	1
1941	5	8	12	1
1942	5	9	12	1
1943	5	10	12	1
1944	5	11	12	1
1945	5	12	12	1
1946	5	13	12	1
1947	5	14	12	1
1948	5	15	12	1
1949	5	16	12	1
1950	5	17	12	0
1951	5	18	12	1
1952	5	19	12	1
1953	5	20	12	1
1954	5	0	13	1
1955	5	1	13	1
1956	5	2	13	0
1957	5	3	13	1
1958	5	4	13	1
1959	5	5	13	1

1960	5	6	13	1
1961	5	7	13	1
1962	5	8	13	1
1963	5	9	13	0
1964	5	10	13	1
1965	5	11	13	1
1966	5	12	13	1
1967	5	13	13	1
1968	5	14	13	1
1969	5	15	13	1
1970	5	16	13	1
1971	5	17	13	0
1972	5	18	13	1
1973	5	19	13	1
1974	5	20	13	1
1975	5	0	14	1
1976	5	1	14	1
1977	5	2	14	0
1978	5	3	14	1
1979	5	4	14	1
1980	5	5	14	1
1981	5	6	14	1
1982	5	7	14	1
1983	5	8	14	1
1984	5	9	14	0
1985	5	10	14	1
1986	5	11	14	1
1987	5	12	14	1
1988	5	13	14	1
1989	5	14	14	1
1990	5	15	14	1
1991	5	16	14	1
1992	5	17	14	0
1993	5	18	14	1
1994	5	19	14	1
1995	5	20	14	1
1996	5	0	15	1
1997	5	1	15	1
1998	5	2	15	0
1999	5	3	15	1

2000	5	4	15	1
2001	5	5	15	1
2002	5	6	15	1
2003	5	7	15	1
2004	5	8	15	1
2005	5	9	15	0
2006	5	10	15	1
2007	5	11	15	1
2008	5	12	15	1
2009	5	13	15	1
2010	5	14	15	1
2011	5	15	15	1
2012	5	16	15	1
2013	5	17	15	0
2014	5	18	15	1
2015	5	19	15	1
2016	5	20	15	1
2017	5	0	16	1
2018	5	1	16	1
2019	5	2	16	0
2020	5	3	16	1
2021	5	4	16	1
2022	5	5	16	1
2023	5	6	16	1
2024	5	7	16	1
2025	5	8	16	1
2026	5	9	16	0
2027	5	10	16	0
2028	5	11	16	0
2029	5	12	16	0
2030	5	13	16	0
2031	5	14	16	0
2032	5	15	16	0
2033	5	16	16	0
2034	5	17	16	0
2035	5	18	16	1
2036	5	19	16	1
2037	5	20	16	1
2038	5	0	17	1
2039	5	1	17	1

2040	5	2	17	0
2041	5	3	17	1
2042	5	4	17	1
2043	5	5	17	1
2044	5	6	17	1
2045	5	7	17	1
2046	5	8	17	1
2047	5	9	17	1
2048	5	10	17	1
2049	5	11	17	1
2050	5	12	17	1
2051	5	13	17	1
2052	5	14	17	1
2053	5	15	17	1
2054	5	16	17	1
2055	5	17	17	1
2056	5	18	17	1
2057	5	19	17	1
2058	5	20	17	1
2059	5	0	18	1
2060	5	1	18	1
2061	5	2	18	0
2062	5	3	18	0
2063	5	4	18	0
2064	5	5	18	0
2065	5	6	18	0
2066	5	7	18	0
2067	5	8	18	0
2068	5	9	18	0
2069	5	10	18	1
2070	5	11	18	1
2071	5	12	18	1
2072	5	13	18	1
2073	5	14	18	1
2074	5	15	18	1
2075	5	16	18	1
2076	5	17	18	1
2077	5	18	18	1
2078	5	19	18	1
2079	5	20	18	1

2080	5	0	19	1
2081	5	1	19	1
2082	5	2	19	1
2083	5	3	19	1
2084	5	4	19	1
2085	5	5	19	1
2086	5	6	19	1
2087	5	7	19	1
2088	5	8	19	1
2089	5	9	19	1
2090	5	10	19	1
2091	5	11	19	1
2092	5	12	19	1
2093	5	13	19	1
2094	5	14	19	1
2095	5	15	19	1
2096	5	16	19	1
2097	5	17	19	1
2098	5	18	19	1
2099	5	19	19	1
2100	5	20	19	1

locationToVisit - table test

The location in building 1 floor 1 is as follow

Coordinate of location 1 and location 2:

- location 1: (9,9)
- location 2: (9,11)
- location 3: (9,17)

```
floor = 1
x = 9
y= 17
sql = 'select * from'\
' (select * from floorMapPoints fmp'\
' where fmp.idfloorMap = ' + str(floor) + ') fp'\
' where fp.x = ' + str(x) + ' and fp.y = '+ str(y)

print(sql)
df = pd.read_sql(sql,connection)['coordinatePoint'][0]
df
```

```
select * from (select * from floorMapPoints fmp where fmp.idfloorMap = 1) fp where fp.x = 9 and fp.y
= 17
```

```
367
```

```
# Generate locations for building 1 floor 1
def generate_location_to_visited(floor=[], locations_coordinate=[]):
    result = []
    id_location = 1
    for floor_id in floor:
        for location_coordinate in locations_coordinate:
            floor = floor_id
            x = location_coordinate[0]
            y = location_coordinate[1]
            location_no = len(locations_coordinate) if id_location % len(locations_coordinate)==0
else id_location % len(locations_coordinate)

            sql = 'select * from'\
            ' (select * from floorMapPoints fmp'\
            ' where fmp.idfloorMap = ' + str(floor) + ') fp'\
            ' where fp.x = ' + str(x) + ' and fp.y = '+ str(y)
            map_point_id = pd.read_sql(sql,connection)['coordinatePoint'][0]


result.append([id_location,map_point_id,'floor_'+str(floor)+'_location_'+str(location_no),'floor_'+s
tr(floor)+'_location_'+str(location_no)+'_description'])
            id_location+=1

    return result
```

```
location_to_visited_data = generate_location_to_visited(floor=[1,2,3,4,5],locations_coordinate=
[(9,9),(9,11),(9,17)])
```

```
location_to_visited_columns = ['idlocationToVisiti','coordinatePoint','name','description']
location_to_visited =
pd.DataFrame(data=location_to_visited_data,columns=location_to_visited_columns)
```

```
pd.io.sql.to_sql(location_to_visited,'locationToVisited',connection,schema='mydb',if_exists='append'
,index=False)
```

```
/Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/pandas/io/sql.py:1336: UserWarning: The
provided table name 'locationToVisited' is not found exactly as such in the database after writing
the table, possibly due to case sensitivity issues. Consider using lower case table names.
  warnings.warn(msg, UserWarning)
```

```
%%sql
select * from locationToVisited
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
15 rows affected.
```

idlocationToVisiti	coordinatePoint	name	description
1	199	floor_1_location_1	floor_1_location_1_description

2	241	floor_1_location_2	floor_1_location_2_description
3	367	floor_1_location_3	floor_1_location_3_description
4	619	floor_2_location_1	floor_2_location_1_description
5	661	floor_2_location_2	floor_2_location_2_description
6	787	floor_2_location_3	floor_2_location_3_description
7	1039	floor_3_location_1	floor_3_location_1_description
8	1081	floor_3_location_2	floor_3_location_2_description
9	1207	floor_3_location_3	floor_3_location_3_description
10	1459	floor_4_location_1	floor_4_location_1_description
11	1501	floor_4_location_2	floor_4_location_2_description
12	1627	floor_4_location_3	floor_4_location_3_description
13	1879	floor_5_location_1	floor_5_location_1_description
14	1921	floor_5_location_2	floor_5_location_2_description
15	2047	floor_5_location_3	floor_5_location_3_description

locationToChangeFloor - table test

Building 1 floor 1 2 3

In building 1:

- elevator: (13,4)
- start: (13,14)

Building 2 floor 1 2

In building 2:

- elevator: (4,16)
- stair: (4,4)

```
# Generate locationToChangeFloor for building 1 floor 1
def generate_location_to_change_floor(floor=[], name=[], locations_coordinate=[]):
    result = []
    id_location = 1
    for floor_id in floor:
        for location_coordinate in locations_coordinate:
            floor = floor_id
            x = location_coordinate[0]
            y = location_coordinate[1]
            location_no = len(locations_coordinate) if id_location % len(locations_coordinate)==0
else id_location % len(locations_coordinate)
            location_type = name[location_no-1]
            is_friendly = True if name[location_no-1] == 'elevator' else False
            timeCost = 5 if is_friendly else 10

            sql = 'select * from'\
            ' (select * from floorMapPoints fmp'\
            ' where fmp.idfloorMap = ' + str(floor) + ') fp'\
            ' where fp.x = ' + str(x) + ' and fp.y = '+ str(y)
            map_point_id = pd.read_sql(sql,connection)['coordinatePoint'][0]

            result.append([id_location,
                           map_point_id,
```

```
'floor_'+str(floor)+'_location_to_change_floor_'+str(location_no)+'_'+name[location_no-1],
                            location_type,
                            is_friendly,
                            timeCost])
            id_location+=1

    return result
```

```
location_to_change_floor_data = generate_location_to_change_floor(floor=[1,2,3],name=
['elevator','stair'], locations_coordinate=[(13,4),(13,14)])
```

```
location_to_change_floor_columns =
['idlocationToChangeFloor','coordinatePoint','name','type','isdisabilityFriendly','distance']
location_to_change_floor =
pd.DataFrame(data=location_to_change_floor_data,columns=location_to_change_floor_columns)
location_to_change_floor
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	idlocationToChangeFloor	coordinatePoint	name	type	isdisabilityFriendly	distance
0	1	98	floor_1_location_to_change_floor_1_elevator	elevator	True	5
1	2	308	floor_1_location_to_change_floor_2_stair	stair	False	10
2	3	518	floor_2_location_to_change_floor_1_elevator	elevator	True	5
3	4	728	floor_2_location_to_change_floor_2_stair	stair	False	10
4	5	938	floor_3_location_to_change_floor_1_elevator	elevator	True	5
5	6	1148	floor_3_location_to_change_floor_2_stair	stair	False	10

```
pd.io.sql.to_sql(location_to_change_floor,'locationToChangeFloor',connection,schema='mydb',if_exists
='append',index=False)
```

```
/Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/pandas/io/sql.py:1336: UserWarning: The
provided table name 'locationToChangeFloor' is not found exactly as such in the database after
writing the table, possibly due to case sensitivity issues. Consider using lower case table names.
  warnings.warn(msg, UserWarning)
```

```
%%sql
select * from locationToChangeFloor
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
6 rows affected.
```

idlocationToChangeFloor	coordinatePoint	name	type	isdisabilityFriendly	distance
1	98	floor_1_location_to_change_floor_1_elevator	elevator	1	5
2	308	floor_1_location_to_change_floor_2_stair	stair	0	10
3	518	floor_2_location_to_change_floor_1_elevator	elevator	1	5
4	728	floor_2_location_to_change_floor_2_stair	stair	0	10
5	938	floor_3_location_to_change_floor_1_elevator	elevator	1	5
6	1148	floor_3_location_to_change_floor_2_stair	stair	0	10

beacons - table test

These are beacons positions inside floors

I assume all floors use the same pattern

The position:

- (4,4)
- (7,5)
- (4,7)
- (7,8)
- (11,3)
- (15,3)
- (11,6)
- (11,9)
- (11,12)
- (15,11)
- (15,14)

```
# Generate beacon for all floor
def generate_beacon_location(floor=[], beacons_locations_coordinate=[]):
    result = []
    id_location = 1
    for floor_id in floor:
        for location_coordinate in beacons_locations_coordinate:
            floor = floor_id
            x = location_coordinate[0]
            y = location_coordinate[1]
            location_no = len(beacons_locations_coordinate) if id_location %
len(beacons_locations_coordinate)==0 else id_location % len(beacons_locations_coordinate)

            sql = 'select * from'\
            ' (select * from floorMapPoints fmp'\
            ' where fmp.idfloorMap = ' + str(floor) + ') fp'\
            ' where fp.x = ' + str(x) + ' and fp.y = '+ str(y)
            map_point_id = pd.read_sql(sql,connection)['coordinatePoint'][0]

            result.append([id_location,
                           map_point_id,
                           'floor_'+str(floor)+'_beacon_model_'+str(location_no),
                           'floor_'+str(floor)+'_beacon_'+str(location_no)+'_description',
                           True])
            id_location+=1

    return result
```

```
beacon_location_data = generate_beacon_location(floor=[1,2,3,4,5],beacons_locations_coordinate=
[(4,4),(7,5),(4,7),(7,8),(11,3),(15,3),(11,6),(11,9),(11,12),(15,11),(15,14)])
```

```
beacon_location_columns = ['idbeacon','coordinatePoint','beaconModel','description','ifwork']
beacon_location = pd.DataFrame(data=beacon_location_data,columns=beacon_location_columns)
```

```
pd.io.sql.to_sql(beacon_location,'beacons',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from beacons
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
55 rows affected.
```

idbeacon	coordinatePoint	beaconModel	description	ifwork
1	89	floor_1_beacon_model_1	floor_1_beacon_1_description	1
2	113	floor_1_beacon_model_2	floor_1_beacon_2_description	1
3	152	floor_1_beacon_model_3	floor_1_beacon_3_description	1
4	176	floor_1_beacon_model_4	floor_1_beacon_4_description	1
5	75	floor_1_beacon_model_5	floor_1_beacon_5_description	1
6	79	floor_1_beacon_model_6	floor_1_beacon_6_description	1
7	138	floor_1_beacon_model_7	floor_1_beacon_7_description	1
8	201	floor_1_beacon_model_8	floor_1_beacon_8_description	1
9	264	floor_1_beacon_model_9	floor_1_beacon_9_description	1
10	247	floor_1_beacon_model_10	floor_1_beacon_10_description	1
11	310	floor_1_beacon_model_11	floor_1_beacon_11_description	1
12	509	floor_2_beacon_model_1	floor_2_beacon_1_description	1
13	533	floor_2_beacon_model_2	floor_2_beacon_2_description	1
14	572	floor_2_beacon_model_3	floor_2_beacon_3_description	1
15	596	floor_2_beacon_model_4	floor_2_beacon_4_description	1
16	495	floor_2_beacon_model_5	floor_2_beacon_5_description	1
17	499	floor_2_beacon_model_6	floor_2_beacon_6_description	1
18	558	floor_2_beacon_model_7	floor_2_beacon_7_description	1
19	621	floor_2_beacon_model_8	floor_2_beacon_8_description	1
20	684	floor_2_beacon_model_9	floor_2_beacon_9_description	1
21	667	floor_2_beacon_model_10	floor_2_beacon_10_description	1
22	730	floor_2_beacon_model_11	floor_2_beacon_11_description	1
23	929	floor_3_beacon_model_1	floor_3_beacon_1_description	1
24	953	floor_3_beacon_model_2	floor_3_beacon_2_description	1
25	992	floor_3_beacon_model_3	floor_3_beacon_3_description	1
26	1016	floor_3_beacon_model_4	floor_3_beacon_4_description	1
27	915	floor_3_beacon_model_5	floor_3_beacon_5_description	1
28	919	floor_3_beacon_model_6	floor_3_beacon_6_description	1

29	978	floor_3_beacon_model_7	floor_3_beacon_7_description	1
30	1041	floor_3_beacon_model_8	floor_3_beacon_8_description	1
31	1104	floor_3_beacon_model_9	floor_3_beacon_9_description	1
32	1087	floor_3_beacon_model_10	floor_3_beacon_10_description	1
33	1150	floor_3_beacon_model_11	floor_3_beacon_11_description	1
34	1349	floor_4_beacon_model_1	floor_4_beacon_1_description	1
35	1373	floor_4_beacon_model_2	floor_4_beacon_2_description	1
36	1412	floor_4_beacon_model_3	floor_4_beacon_3_description	1
37	1436	floor_4_beacon_model_4	floor_4_beacon_4_description	1
38	1335	floor_4_beacon_model_5	floor_4_beacon_5_description	1
39	1339	floor_4_beacon_model_6	floor_4_beacon_6_description	1
40	1398	floor_4_beacon_model_7	floor_4_beacon_7_description	1
41	1461	floor_4_beacon_model_8	floor_4_beacon_8_description	1
42	1524	floor_4_beacon_model_9	floor_4_beacon_9_description	1
43	1507	floor_4_beacon_model_10	floor_4_beacon_10_description	1
44	1570	floor_4_beacon_model_11	floor_4_beacon_11_description	1
45	1769	floor_5_beacon_model_1	floor_5_beacon_1_description	1
46	1793	floor_5_beacon_model_2	floor_5_beacon_2_description	1
47	1832	floor_5_beacon_model_3	floor_5_beacon_3_description	1
48	1856	floor_5_beacon_model_4	floor_5_beacon_4_description	1
49	1755	floor_5_beacon_model_5	floor_5_beacon_5_description	1
50	1759	floor_5_beacon_model_6	floor_5_beacon_6_description	1
51	1818	floor_5_beacon_model_7	floor_5_beacon_7_description	1
52	1881	floor_5_beacon_model_8	floor_5_beacon_8_description	1
53	1944	floor_5_beacon_model_9	floor_5_beacon_9_description	1
54	1927	floor_5_beacon_model_10	floor_5_beacon_10_description	1
55	1990	floor_5_beacon_model_11	floor_5_beacon_11_description	1

User information simulation

- adminUser
- customer

```
# Add admin user data
admin_user_data = [[1,'admin','admin']]
```

```
admin_user_columns = ['idadminUser','name','password']
admin_user = pd.DataFrame(data = admin_user_data,columns=admin_user_columns)
```

```
pd.io.sql.to_sql(admin_user,'adminUser',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from adminUser
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
1 rows affected.
```

idadminUser	name	password
1	admin	admin

```
# Add customer user data
customer_user_data = []
for i in range(10):
    customer_user_data.append([
        i+1,
        'user_name_'+str(i),
        'password_'+str(i),
        'first_name_'+str(i),
        'last_name_'+str(i),
        False
    ])
customer_user_data.append([
    11,
    'disability_1',
    'password_'+str(i),
    'first_name_'+str(i),
    'last_name_'+str(i),
    True
])
```

```
customer_user_columns = ['idcustomer','userName','password','firstName','lastName','isdisability']
customer_user = pd.DataFrame(data = customer_user_data,columns=customer_user_columns)
```

```
pd.io.sql.to_sql(customer_user,'customer',connection,schema='mydb',if_exists='append',index=False)
```

```
%%sql
select * from customer
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
11 rows affected.
```

idcustomer	userName	password	firstName	lastName	isdisability
1	user_name_0	password_0	first_name_0	last_name_0	0
2	user_name_1	password_1	first_name_1	last_name_1	0
3	user_name_2	password_2	first_name_2	last_name_2	0
4	user_name_3	password_3	first_name_3	last_name_3	0
5	user_name_4	password_4	first_name_4	last_name_4	0
6	user_name_5	password_5	first_name_5	last_name_5	0
7	user_name_6	password_6	first_name_6	last_name_6	0

8	user_name_7	password_7	first_name_7	last_name_7	0
9	user_name_8	password_8	first_name_8	last_name_8	0
10	user_name_9	password_9	first_name_9	last_name_9	0
11	disability_1	password_9	first_name_9	last_name_9	1

Positioning test

```
# Generate a device for disability_1
customer_device_data = [
    [1,11,'iphone',0]
]
```

```
customer_device_columns = ['iddevice','idcustomer','deviceTpye','NumConnectionsAvailable']
customer_device = pd.DataFrame(data = customer_device_data,columns=customer_device_columns)
```

```
pd.io.sql.to_sql(customer_device,'device',connection,schema='mydb',if_exists='append',index=False)
```

```
/Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py:170: Warning: (3719,
"'utf8' is currently an alias for the character set UTF8MB3, but will be an alias for UTF8MB4 in a
future release. Please consider using UTF8MB4 in order to be unambiguous.")
  result = self._query(query)
```

```
%%sql
select * from device
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
1 rows affected.
```

iddevice	idcustomer	deviceTpye	NumConnectionsAvailable
1	11	iphone	0

```
%%sql
# Select coordinatePoint in a particular building floor
select fmp.coordinatePoint, fmp.x,fmp.y  from floorMap flm
join floorMapPoints fmp
on flm.idfloorMap = fmp.idfloorMap
where flm.idbuilding = 1 and flm.floorNumber = 1
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
420 rows affected.
```

coordinatePoint	x	y
1	0	0
2	1	0
3	2	0

4	3	0
5	4	0
6	5	0
7	6	0
8	7	0
9	8	0
10	9	0
11	10	0
12	11	0
13	12	0
14	13	0
15	14	0
16	15	0
17	16	0
18	17	0
19	18	0
20	19	0
21	20	0
22	0	1
23	1	1
24	2	1
25	3	1
26	4	1
27	5	1
28	6	1
29	7	1
30	8	1
31	9	1
32	10	1
33	11	1
34	12	1
35	13	1
36	14	1
37	15	1
38	16	1
39	17	1
40	18	1
41	19	1
42	20	1
43	0	2

44	1	2
45	2	2
46	3	2
47	4	2
48	5	2
49	6	2
50	7	2
51	8	2
52	9	2
53	10	2
54	11	2
55	12	2
56	13	2
57	14	2
58	15	2
59	16	2
60	17	2
61	18	2
62	19	2
63	20	2
64	0	3
65	1	3
66	2	3
67	3	3
68	4	3
69	5	3
70	6	3
71	7	3
72	8	3
73	9	3
74	10	3
75	11	3
76	12	3
77	13	3
78	14	3
79	15	3
80	16	3
81	17	3
82	18	3
83	19	3

84	20	3
85	0	4
86	1	4
87	2	4
88	3	4
89	4	4
90	5	4
91	6	4
92	7	4
93	8	4
94	9	4
95	10	4
96	11	4
97	12	4
98	13	4
99	14	4
100	15	4
101	16	4
102	17	4
103	18	4
104	19	4
105	20	4
106	0	5
107	1	5
108	2	5
109	3	5
110	4	5
111	5	5
112	6	5
113	7	5
114	8	5
115	9	5
116	10	5
117	11	5
118	12	5
119	13	5
120	14	5
121	15	5
122	16	5
123	17	5

124	18	5
125	19	5
126	20	5
127	0	6
128	1	6
129	2	6
130	3	6
131	4	6
132	5	6
133	6	6
134	7	6
135	8	6
136	9	6
137	10	6
138	11	6
139	12	6
140	13	6
141	14	6
142	15	6
143	16	6
144	17	6
145	18	6
146	19	6
147	20	6
148	0	7
149	1	7
150	2	7
151	3	7
152	4	7
153	5	7
154	6	7
155	7	7
156	8	7
157	9	7
158	10	7
159	11	7
160	12	7
161	13	7
162	14	7
163	15	7

164	16	7
165	17	7
166	18	7
167	19	7
168	20	7
169	0	8
170	1	8
171	2	8
172	3	8
173	4	8
174	5	8
175	6	8
176	7	8
177	8	8
178	9	8
179	10	8
180	11	8
181	12	8
182	13	8
183	14	8
184	15	8
185	16	8
186	17	8
187	18	8
188	19	8
189	20	8
190	0	9
191	1	9
192	2	9
193	3	9
194	4	9
195	5	9
196	6	9
197	7	9
198	8	9
199	9	9
200	10	9
201	11	9
202	12	9
203	13	9

204	14	9
205	15	9
206	16	9
207	17	9
208	18	9
209	19	9
210	20	9
211	0	10
212	1	10
213	2	10
214	3	10
215	4	10
216	5	10
217	6	10
218	7	10
219	8	10
220	9	10
221	10	10
222	11	10
223	12	10
224	13	10
225	14	10
226	15	10
227	16	10
228	17	10
229	18	10
230	19	10
231	20	10
232	0	11
233	1	11
234	2	11
235	3	11
236	4	11
237	5	11
238	6	11
239	7	11
240	8	11
241	9	11
242	10	11
243	11	11

244	12	11
245	13	11
246	14	11
247	15	11
248	16	11
249	17	11
250	18	11
251	19	11
252	20	11
253	0	12
254	1	12
255	2	12
256	3	12
257	4	12
258	5	12
259	6	12
260	7	12
261	8	12
262	9	12
263	10	12
264	11	12
265	12	12
266	13	12
267	14	12
268	15	12
269	16	12
270	17	12
271	18	12
272	19	12
273	20	12
274	0	13
275	1	13
276	2	13
277	3	13
278	4	13
279	5	13
280	6	13
281	7	13
282	8	13
283	9	13

284	10	13
285	11	13
286	12	13
287	13	13
288	14	13
289	15	13
290	16	13
291	17	13
292	18	13
293	19	13
294	20	13
295	0	14
296	1	14
297	2	14
298	3	14
299	4	14
300	5	14
301	6	14
302	7	14
303	8	14
304	9	14
305	10	14
306	11	14
307	12	14
308	13	14
309	14	14
310	15	14
311	16	14
312	17	14
313	18	14
314	19	14
315	20	14
316	0	15
317	1	15
318	2	15
319	3	15
320	4	15
321	5	15
322	6	15
323	7	15

324	8	15
325	9	15
326	10	15
327	11	15
328	12	15
329	13	15
330	14	15
331	15	15
332	16	15
333	17	15
334	18	15
335	19	15
336	20	15
337	0	16
338	1	16
339	2	16
340	3	16
341	4	16
342	5	16
343	6	16
344	7	16
345	8	16
346	9	16
347	10	16
348	11	16
349	12	16
350	13	16
351	14	16
352	15	16
353	16	16
354	17	16
355	18	16
356	19	16
357	20	16
358	0	17
359	1	17
360	2	17
361	3	17
362	4	17
363	5	17

364	6	17
365	7	17
366	8	17
367	9	17
368	10	17
369	11	17
370	12	17
371	13	17
372	14	17
373	15	17
374	16	17
375	17	17
376	18	17
377	19	17
378	20	17
379	0	18
380	1	18
381	2	18
382	3	18
383	4	18
384	5	18
385	6	18
386	7	18
387	8	18
388	9	18
389	10	18
390	11	18
391	12	18
392	13	18
393	14	18
394	15	18
395	16	18
396	17	18
397	18	18
398	19	18
399	20	18
400	0	19
401	1	19
402	2	19
403	3	19

404	4	19
405	5	19
406	6	19
407	7	19
408	8	19
409	9	19
410	10	19
411	11	19
412	12	19
413	13	19
414	14	19
415	15	19
416	16	19
417	17	19
418	18	19
419	19	19
420	20	19

```
%%sql
select * from beacons b
join
# Select coordinatePoint in a particular building floor
(select fmp.coordinatePoint, fmp.x,fmp.y  from floorMap flm
join floorMapPoints fmp
on flm.idfloorMap = fmp.idfloorMap
where flm.idbuilding = 1 and flm.floorNumber = 1) bfp
on b.coordinatePoint = bfp.coordinatePoint
```

```
 * mysql+pymysql://brickea_mac:***@rm-0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
11 rows affected.
```

idbeacon	coordinatePoint	beaconModel	description	ifwork	coordinatePoint_1	x	y
1	89	floor_1_beacon_model_1	floor_1_beacon_1_description	1	89	4	4
2	113	floor_1_beacon_model_2	floor_1_beacon_2_description	1	113	7	5
3	152	floor_1_beacon_model_3	floor_1_beacon_3_description	1	152	4	7
4	176	floor_1_beacon_model_4	floor_1_beacon_4_description	1	176	7	8
5	75	floor_1_beacon_model_5	floor_1_beacon_5_description	1	75	11	3
6	79	floor_1_beacon_model_6	floor_1_beacon_6_description	1	79	15	3
7	138	floor_1_beacon_model_7	floor_1_beacon_7_description	1	138	11	6
8	201	floor_1_beacon_model_8	floor_1_beacon_8_description	1	201	11	9
9	264	floor_1_beacon_model_9	floor_1_beacon_9_description	1	264	11	12
10	247	floor_1_beacon_model_10	floor_1_beacon_10_description	1	247	15	11
11	310	floor_1_beacon_model_11	floor_1_beacon_11_description	1	310	15	14

```
# Get beacon data from a particular building floor
building = 1
floor = 1
sql = 'select * from beacons b'\
        ' join'\
        ' (select fmp.coordinatePoint, fmp.x,fmp.y  from floorMap flm'\
        ' join floorMapPoints fmp'\
        ' on flm.idfloorMap = fmp.idfloorMap'\
        ' where flm.idbuilding = '+str(building)+' and flm.floorNumber = '+str(floor)+') bfp'\
        ' on b.coordinatePoint = bfp.coordinatePoint'
beacon_points = pd.read_sql(sql,connection)
beacon_points
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	idbeacon	coordinatePoint	beaconModel	description	ifwork	coordinatePoint	x	y
0	1	89	floor_1_beacon_model_1	floor_1_beacon_1_description	1	89	4.0	4.0
1	2	113	floor_1_beacon_model_2	floor_1_beacon_2_description	1	113	7.0	5.0
2	3	152	floor_1_beacon_model_3	floor_1_beacon_3_description	1	152	4.0	7.0
3	4	176	floor_1_beacon_model_4	floor_1_beacon_4_description	1	176	7.0	8.0
4	5	75	floor_1_beacon_model_5	floor_1_beacon_5_description	1	75	11.0	3.0
5	6	79	floor_1_beacon_model_6	floor_1_beacon_6_description	1	79	15.0	3.0
6	7	138	floor_1_beacon_model_7	floor_1_beacon_7_description	1	138	11.0	6.0
7	8	201	floor_1_beacon_model_8	floor_1_beacon_8_description	1	201	11.0	9.0
8	9	264	floor_1_beacon_model_9	floor_1_beacon_9_description	1	264	11.0	12.0
9	10	247	floor_1_beacon_model_10	floor_1_beacon_10_description	1	247	15.0	11.0
10	11	310	floor_1_beacon_model_11	floor_1_beacon_11_description	1	310	15.0	14.0

```
# Generate random 4 connections between device and beacon
simulated_4_beacons = beacon_points.sample(n=4)
simulated_4_beacons
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	idbeacon	coordinatePoint	beaconModel	description	ifwork	coordinatePoint	x	y
4	5	75	floor_1_beacon_model_5	floor_1_beacon_5_description	1	75	11.0	3.0
1	2	113	floor_1_beacon_model_2	floor_1_beacon_2_description	1	113	7.0	5.0

	idbeacon	coordinatePoint	beaconModel	description	ifwork	coordinatePoint	x	y
6	7	138	floor_1_beacon_model_7	floor_1_beacon_7_description	1	138	11.0	6.0
8	9	264	floor_1_beacon_model_9	floor_1_beacon_9_description	1	264	11.0	12.0

```
# Get floor map points for a particular building floor
sql = 'select fmp.coordinatePoint, fmp.x,fmp.y,fmp.ifpassable from floorMap flm'\
        ' join floorMapPoints fmp'\
        ' on flm.idfloorMap = fmp.idfloorMap'\
        ' where flm.idbuilding = '+str(building)+' and flm.floorNumber = '+str(floor)
floor_points = pd.read_sql(sql,connection)
floor_points
```

```
/Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/pymysql/cursors.py:170: Warning: (3719,
"'utf8' is currently an alias for the character set UTF8MB3, but will be an alias for UTF8MB4 in a
future release. Please consider using UTF8MB4 in order to be unambiguous.")
  result = self._query(query)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	coordinatePoint	x	y	ifpassable
0	1	0.0	0.0	1
1	2	1.0	0.0	1
2	3	2.0	0.0	1
3	4	3.0	0.0	1
4	5	4.0	0.0	1
...	...	...	...	...
415	416	16.0	19.0	1
416	417	17.0	19.0	1
417	418	18.0	19.0	1
418	419	19.0	19.0	1
419	420	20.0	19.0	1

420 rows × 4 columns

```
# Generate random position for device
simulated_device_current_point = floor_points.sample(n=1)
while simulated_device_current_point.ifpassable.item() == 0:
    # The random position for device must be passable
    simulated_device_current_point = floor_points.sample(n=1)
simulated_device_current_point
```

```
/Users/brickeawang/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: FutureWarning:
`item` has been deprecated and will be removed in a future version
```

```
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	coordinatePoint	x	y	ifpassable
21	22	0.0	1.0	1

```
# Calculate simulated distance between device and beacons
beacon_device_distance = []
for i in range(len(simulated_4_beacons)):
    distance = np.sqrt(np.square(simulated_device_current_point.iloc[0].x -
simulated_4_beacons.iloc[i].x)
                       + np.square(simulated_device_current_point.iloc[0].y -
simulated_4_beacons.iloc[i].y))
    beacon_device_distance.append((distance))
beacon_device_distance
```

```
[11.180339887498949, 8.06225774829855, 12.083045973594572, 15.556349186104045]
```

```
# Trilateration
def trilateration(beacons=None,distance=None):
    A = 2*(beacons.iloc[2].y - beacons.iloc[0].y)
    A_ = 2*(beacons.iloc[1].y - beacons.iloc[0].y)
    B = 2*(beacons.iloc[2].x - beacons.iloc[0].y)
    B_ = 2*(beacons.iloc[1].x - beacons.iloc[0].y)

    delta_1 = np.square(distance[0]) - np.square(distance[2]) + np.square(beacons.iloc[2].y) -
np.square(beacons.iloc[0].y) + np.square(beacons.iloc[2].x) - np.square(beacons.iloc[0].x)
    delta_2 = np.square(distance[0]) - np.square(distance[1]) + np.square(beacons.iloc[1].y) -
np.square(beacons.iloc[0].y) + np.square(beacons.iloc[1].x) - np.square(beacons.iloc[0].x)

    device_x = (delta_1 * A_ - delta_2 * A)/(B * A_ - B_ * A)
    device_y = (delta_1 * B_ - delta_2 * B)/(B_ * A - B * A_)

    return (round(device_x),round(device_y))
```

```
simulated_4_beacons.iloc[:3]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	idbeacon	coordinatePoint	beaconModel	description	ifwork	coordinatePoint	x	y

	idbeacon	coordinatePoint	beaconModel	description	ifwork	coordinatePoint	x	y
4	5	75	floor_1_beacon_model_5	floor_1_beacon_5_description	1	75	11.0	3.0
1	2	113	floor_1_beacon_model_2	floor_1_beacon_2_description	1	113	7.0	5.0
6	7	138	floor_1_beacon_model_7	floor_1_beacon_7_description	1	138	11.0	6.0

```
trilateration(beacons = simulated_4_beacons.iloc[:3],distance = beacon_device_distance[:3])
```

```
(-0.0, 1.0)
```

```

```