Name: Khushi Nitinkumar Patel

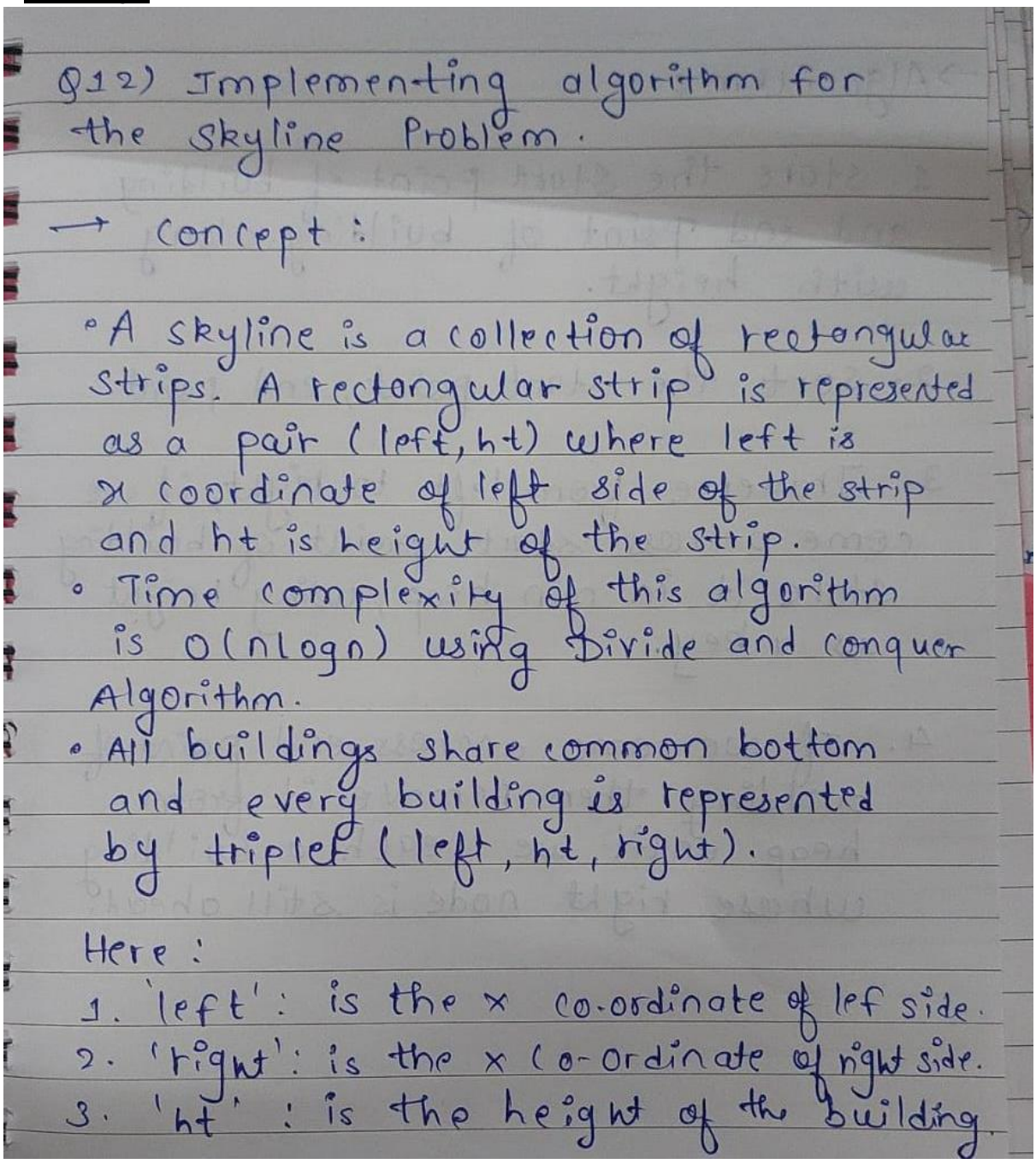
PRN: 2020BTECS00037

Batch: T5

DAA LAB ESE

Q12) Implementing algorithm for The Skyline Problem.

>**Theory:**



Q12) Implementing algorithm for the Skyline Problem.

→ Concept :

- A skyline is a collection of rectangular strips. A rectangular strip is represented as a pair (left, ht) where left is x coordinate of left side of the strip and ht is height of the strip.
- Time complexity of this algorithm is O(nlogn) using Divide and conquer Algorithm.
- All buildings share common bottom and every building is represented by triplet (left, ht, right).

Here :

1. 'left' : is the x co-ordinate of lef side.
2. 'right' : is the x (o-ordinate of right side.
3. 'ht' : is the height of the building.

>**Algorithm:**

Date: / /

→Algorithm:

1. store the start point of building and end point of building along with height.

2. sort the start point, end point.

3. Traverse from left to right, if we come across start point of building store it in min heap, using height as a key.

4. If we come across end point of building then remove it from heap until we reach a building whose right node is still ahead.

MATRIKAS

>**Code:**

```
#include <bits/stdc++.h>
using namespace std;

vector<vector<int>>
getskyline(vector<vector<int>> &buildings)
{
    vector<vector<int>> edges;
    for (int i = 0; i < buildings.size(); i++)
    {
        int x = edges.size();
        edges.push_back(vector<int>());
        edges[x].push_back(buildings[i][0]);
        edges[x].push_back(-buildings[i][2]);
        edges[x].push_back(buildings[i][1]);
    }
    for (int i = 0; i < buildings.size(); i++)
    {
        int x = edges.size();
        edges.push_back(vector<int>());
        edges[x].push_back(buildings[i][1]);
        edges[x].push_back(0);
        edges[x].push_back(1e9);
    }

    sort(edges.begin(),edges.end());

    priority_queue<pair<int,int>,
    vector<pair<int,int>>,
    greater<pair<int,int>>> prevHighest;
    prevHighest.push({0,1e9});
    vector<vector<int>> skyline;

    for(int i=0;i<edges.size();i++){
        int start= edges[i][0];
        int currHeight= -1 * edges[i][1];
        int end=edges[i][2];


     while(prevHighest.top().second<=start){
      prevHighest.pop();
    }
     if(currHeight>0){
     prevHighest.push({-currHeight,end});
    }

    if(skyline.size()==0){
        skyline.push_back(vector<int>());
```

```
        skyline[0].push_back(start);
        skyline[0].push_back(-prevHighest.top().first);
    }
    else if(skyline.back()[1]!=-prevHighest.top().first){
       int x= skyline.size();
       skyline.push_back(vector<int>());
       skyline[x].push_back(start);
       skyline[x].push_back(-prevHighest.top().first);

        }

     }
     return skyline;
}

int main(){
  vector<vector<int>>
  buildings ={{2,9,10},{3,7,15},{5,12,12},{15,20,10},{19,24,8}};
  vector<vector<int>>ans= getskyline(buildings);
  buildings={{12,16,7},{14,25,3},{19,22,18},{23,29,13},{24,28,4}};

  for(int i=0;i<ans.size();i++){
    cout<<ans[i][0]<<" "<<ans[i][1]<<endl;
  }
  return 0;
}
```

>**Output:**

```
PROBLEMS   OUTPUT   TERMINAL   JUPYTER

PS C:\Users\khush\Desktop\daa_ese> cd "c:\Users\khush\Desktop\daa_ese\" ; if ($?) { g++ sky
2 10
3 15
7 12
12 0
15 10
20 8
24 0
PS C:\Users\khush\Desktop\daa_ese>
```