DAA Assignment 3

Name: Vishal Sule

Roll no.: 234

PRN: 0120190064

Div: D3

Problem Statement: Assume that, there are N Person and each person needs exactly one cab. For each person you are given the start time and end time (both inclusive) during which that person will travel. Find the Minimum number of cans required. Which Strategy will be best suitable to solve this problem?

Code:

```
#include<iostream>
#include <bits/stdc++.h>
#include <algorithm>
#include <chrono>
using namespace std;
using namespace std::chrono;

using namespace std;
//#define N 6                  // defines the number of activities

// Structure represents an activity having start time and finish time.
struct Activity
{
```

```
   int start, finish;
};

// This function is used for sorting activities according to finish time
bool Sort_activity(Activity s1, Activity s2)
{
   return (s1.finish< s2.finish);
}

/*      Prints maximum number of activities that can
        be done by a single person or single machine at a time.
*/
void print_Max_Activities(Activity arr[], int n)
{
   // Sort activities according to finish time
        sort(arr, arr+n, Sort_activity);

        cout<< "Following activities are selected \n";

   // Select the first activity
   int i = 0;
        cout<< "(" <<arr[i].start<< ", " <<arr[i].finish << ")\n";

   // Consider the remaining activities from 1 to n-1
   for (int j = 1; j < n; j++)
{
```

```
        // Select this activity if it has start time greater than or equal
        // to the finish time of previously selected activity
        if (arr[j].start>= arr[i].finish)
        {
                        cout<< "(" <<arr[j].start<< ", "<<arr[j].finish << ") \n";
                        i = j;
        }
    }
}

// Driver program
int main()
{
        int N;
        cout<<"Enter Number of activities : ";
        cin>>N;
    Activity arr[N];
        for(int i=0; i<=N-1; i++)
        {
                cout<<"\nEnter the start and end time of "<<i+1<<" activity \n";
                cin>>arr[i].start>>arr[i].finish;
    }

        auto start = high_resolution_clock::now();

        print_Max_Activities(arr, N);
```

```
        auto stop = high_resolution_clock::now();
        auto duration = duration_cast<nanoseconds>(stop - start);
        cout << "\nTime taken by function: "<< duration.count() << "
nanoseconds";


   return 0;
}
```

Output:

```
Enter userof a tivitiez : 6

Enter the start aha era time of 1 activity
  9

Enter the start aha era time or z activity
1

Enter the start aha era time of ? activity
  4

Enter the start aha era time of 4 activity
0 6

Enter the start aha era time of & activity
  7

Enter the start aha era time of 6 activity
  9
sollowing activities are zelectea
(1, 2)
(3,  4)
(5, 7)
(9,  9)

time taen    fa t1»: iseio   zowcoaa
```