Marwadi University	**Marwari University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	**Date: 13\09\2025**	**Enrollment No: 92301733054**

I. Job Scheduling Problem

Code :-

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

class Job {
public:
    int id;
    int deadline;
    int profit;
    Job(int id, int deadline, int profit): id(id), deadline(deadline), profit(profit) {}
};

pair<int, long long> Job_Scheduling(vector<Job> jobs) {
    long long Max_Profit = 0.0;
    int count_of_jobs = 0;
    int max_deadline = -1;
    sort(jobs.begin(), jobs.end(), [](const Job& a, const Job& b) {
        return a.profit > b.profit;
        });
    for (const Job& job : jobs) {
        max_deadline = max(max_deadline, job.deadline);
    }
    vector<int> selectedJobs(max_deadline + 1, -1);
    for (int i = 0; i < jobs.size(); i++) {
        for (int j = jobs[i].deadline; j >= 0; j--) {
            if (selectedJobs[j] == -1) {
                selectedJobs[j] = jobs[i].id;
                count_of_jobs++;
                Max_Profit += jobs[i].profit;
                break;
            }
        }
    }

    return { count_of_jobs , Max_Profit };
}


int main() {
    vector<Job> jobs;
```

Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	**Date: 13\09\2025**	**Enrollment No: 92301733054**

```
    vector<int> deadline = { 2, 4, 6, 5, 2, 2, 6, 4 };
    vector<int> profit = { 22, 25, 20, 10, 65, 60, 70, 80 };

    for (int i = 0; i < 8; i++) {
        jobs.push_back(Job(i + 1, deadline[i], profit[i]));
    }

    pair<int, long long> Count_and_profit = Job_Scheduling(jobs);

    cout << "We can Perform " << Count_and_profit.first << " and Earn Profit of Rs. " << Count_and_profit.second << " ." << endl;

    return 0;

}
```

Output :-

II. Activity Selection Problem :-

Code :-

```
#include <bits/stdc++.h>
using namespace std;
class Activity {
public:
    int id;
    int start;
    int end;
    Activity(int id , int start, int end) : id(id) , start(start), end(end) {};
};
vector<Activity> Activity_Selection(vector<Activity>& activities) {
    vector<Activity> selected_activities;
    sort(activities.begin(), activities.end(), [](Activity a, Activity b) {
        return a.end < b.end;
        });
    selected_activities.push_back(activities[0]);
    int last_activity_end = selected_activities.back().end;
    for (int i = 1; i < activities.size(); i++) {
        if (activities[i].start >= last_activity_end) {
            selected_activities.push_back(activities[i]);
            last_activity_end = selected_activities.back().end;
        }
    }
    return selected_activities;
}
```

Marwadi University	**Marwari University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing application-based algorithms using Greedy Approach	
Experiment No: 07	**Date: 13\09\2025**	**Enrollment No: 92301733054**

```
int main() {

    vector<Activity> activities;
    vector<int> start_time = { 5,1,3,0,5,8 };
    vector<int> end_time = { 9,2,4,6,7,9 };
    for (int i = 0; i < start_time.size(); i++) {
        activities.push_back(Activity(i+ 1 ,start_time[i], end_time[i]));
    }
    vector<Activity> selected_activities = Activity_Selection(activities);
    cout << "Selected Activities are :- " << endl;
    for (auto activity : selected_activities) {
        cout << "Activity ID :- " << activity.id <<  " Start Time: " << activity.start << ", End Time: " << activity.end << endl;
    }

    return 0;
}
```

Output :-

```
Selected Activities are :-
Activity  :- 5 Start Time: 7, End Time: 1
Activity  :- 2 Start Time: 4, End Time: 5
Activity  :- 3 Start Time: 5, End Time: 6
Activity  :- 6 Start Time: 9, End Time: 8
```

Conclusion:

We learnt in this experiment that greedy algorithms solve the Job Scheduling and Activity Selection problems efficiently. Job scheduling maximizes profit within deadlines, while activity selection maximizes non-overlapping activities.