```
//Returns minimum number of platforms required
void printJobScheduling(Job arr[], int n)
{
        // Sort all jobs according to decreasing order of profit
        sort(arr, arr+n, comparison);
        int result[n];          // To store result (Sequence of jobs)
        bool slot[n];           // To keep track of free time slots

        // Initialize all slots to be free
        for (int i=0; i<n; i++)
                slot[i] = false;

        // Iterate through all given jobs
        for (int i=0; i<n; i++)
        {
                // Find a free slot for this job (Note that we start
                // from the last possible slot)
                for (int j=min(n, arr[i].dead)-1; j>=0; j--)
                {
                  // Free slot found
                  if (slot[j]==false)
                  {
                                result[j] = i;  // Add this job to result
                                slot[j] = true; // Make this slot occupied
                                break;
                  }
                }
        }

        // Print the result
        for (int i=0; i<n; i++)
          if (slot[i])
            cout << arr[result[i]].id << " ";
}
```

```
// A structure to represent a job
struct Job
{
  char id;     // Job Id
  int dead;    // Deadline of job
  int profit;  // Profit if job is over before or on deadline
};

// This function is used for sorting all jobs according to profit
bool comparison(Job a, Job b)
{
    return (a.profit > b.profit);
}
```