|  |  |
| --- | --- |
| **Job Sequencing in deadline in C++** | |
| #include <iostream>  #include <vector>  #include <algorithm>  #include <set>  class Job {  public:  char id;  int deadline;  int profit;  Job(char id, int deadline, int profit) {  this->id = id;  this->deadline = deadline;  this->profit = profit;  }  };  struct JobComparator {  bool operator()(const Job& j1, const Job& j2) {  if (j1.profit != j2.profit)  return j2.profit < j1.profit;  else  return j2.deadline < j1.deadline;  }  };  void printJobScheduling(std::vector<Job>& jobs) {  std::sort(jobs.begin(), jobs.end(), JobComparator());  std::set<int> ts;  for (int i = 0; i < jobs.size(); i++)  ts.insert(i);  for (const auto& job : jobs) {  auto it = ts.upper\_bound(job.deadline - 1);  if (it != ts.begin()) {  --it;  std::cout << job.id << " ";  ts.erase(it);  }  }  }  int main() {  std::vector<Job> jobs = {  Job('a', 2, 100),  Job('b', 1, 19),  Job('c', 2, 27),  Job('d', 1, 25),  Job('e', 3, 15)  };  printJobScheduling(jobs);  std::cout << std::endl;  return 0;  } | Input jobs = {  Job('a', 2, 100),  Job('b', 1, 19),  Job('c', 2, 27),  Job('d', 1, 25),  Job('e', 3, 15)  } ➤ Step 1: Sort Jobs by Profit (Descending), Break Tie with Deadline  | **Job** | **Deadline** | **Profit** | | --- | --- | --- | | a | 2 | 100 | | c | 2 | 27 | | d | 1 | 25 | | b | 1 | 19 | | e | 3 | 15 |   After sorting, order remains the same. ➤ Step 2: Initialize Available Time Slots We simulate time slots using a std::set<int> ts.  ts = { 0, 1, 2, 3, 4 } // these are slot \*indices\*, not actual times.  We only need max\_deadline = 3, so slots {0, 1, 2} are enough, but in the code ts.insert(i) for all jobs is used — let's assume the set size is sufficient. ➤ Step 3: Process Jobs One by One We use upper\_bound(job.deadline - 1) to find the latest available slot before deadline.   | **Job** | **Deadline** | **Profit** | **Find Slot ≤ Deadline - 1** | **Result** | **Scheduled?** | **ts After** | | --- | --- | --- | --- | --- | --- | --- | | a | 2 | 100 | upper\_bound(1) → 2 → step back → 1 | ✅ Use slot 1 | Yes | {0, 2, 3, 4} | | c | 2 | 27 | upper\_bound(1) → 2 → step back → 0 | ✅ Use slot 0 | Yes | {2, 3, 4} | | d | 1 | 25 | upper\_bound(0) → 2 → step back → X | ❌ None available | No | {2, 3, 4} | | b | 1 | 19 | upper\_bound(0) → 2 → step back → X | ❌ None available | No | {2, 3, 4} | | e | 3 | 15 | upper\_bound(2) → 3 → step back → 2 | ✅ Use slot 2 | Yes | {3, 4} |  ✅ Final Output (Jobs Scheduled) Output: a c e |
| a c e | |