3. Job sequencing with deadlines

Algorithm:

1. Data Structure: The code defines a structure `Job` which holds information about each job, including its ID, deadline, and profit.

2. Comparison Function: A comparison function `compare` is defined to sort the jobs based on their profit in non-increasing order using `qsort`.

3. Job Scheduling Function: The function `printJobScheduling` takes an array of jobs and its size as input and prints the sequence of jobs to maximize profit while respecting deadlines.

   a. Sort by Profit: First, the jobs are sorted based on their profit using the comparison function `compare`.

   b. Initialize Arrays: Two arrays are initialized:
   - `result[]`: An array to store the indexes of scheduled jobs.
   - `slot[]`: An array to keep track of available slots.

   c. Schedule Jobs: Jobs are scheduled by iterating through the sorted jobs. For each job, it iterates backward from its deadline and finds the first available slot before the deadline. If a slot is found, the job is assigned to that slot, and the slot is marked as taken.

   d. Print Scheduled Jobs: Finally, it prints the IDs of the scheduled jobs.

4. Main Function:
   - It prompts the user to input the number of jobs.
   - It then takes input for each job's ID, deadline, and profit.
   - After sorting the jobs based on profit, it calls the `printJobScheduling` function to print the scheduled jobs.

Code:

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>

typedef struct
{
```

```
    char id;
    int deadline;
    int profit;
} Job;

// Comparison function for qsort
int compare(const void* a, const void* b)
{
    return ((Job*)b)->profit - ((Job*)a)->profit;
}

void printJobScheduling(Job arr[], int n)
{
    // Step 1: Sort jobs by profit
    qsort(arr, n, sizeof(Job), compare);

    int result[n]; // Array to store the indexes of scheduled jobs
    bool slot[n];  // Array to keep track of available slots

    // Step 2: Initialize arrays
    for (int i = 0; i < n; i++)
        slot[i] = false; // Initially, all slots are available

    // Step 3: Schedule jobs
    for (int i = 0; i < n; i++)
            {
        for (int j = (arr[i].deadline - 1); j >= 0; j--) {
            if (!slot[j]) {
                result[j] = i; // Assign the job to this slot
                slot[j] = true; // Mark the slot as taken
                break;
            }
        }
    }

    // Step 4: Print scheduled jobs
    for (int i = 0; i < n; i++)
            {
        if (slot[i])
            printf("%c ", arr[result[i]].id); // Print the ID of the scheduled job
    }
}

int main()
```

```
{
    int numJobs;
    printf("Enter the number of jobs: ");
    scanf("%d", &numJobs);

    Job jobs[numJobs];
    printf("Enter IDs, deadlines, and profits of jobs:\n");
    for (int i = 0; i < numJobs; i++)
          {
       scanf(" %c %d %d", &jobs[i].id, &jobs[i].deadline, &jobs[i].profit); // Corrected input format
for id
    }

    qsort(jobs, numJobs, sizeof(Job), compare);

    printJobScheduling(jobs, numJobs);

    return 0;
}
```

Input-output: