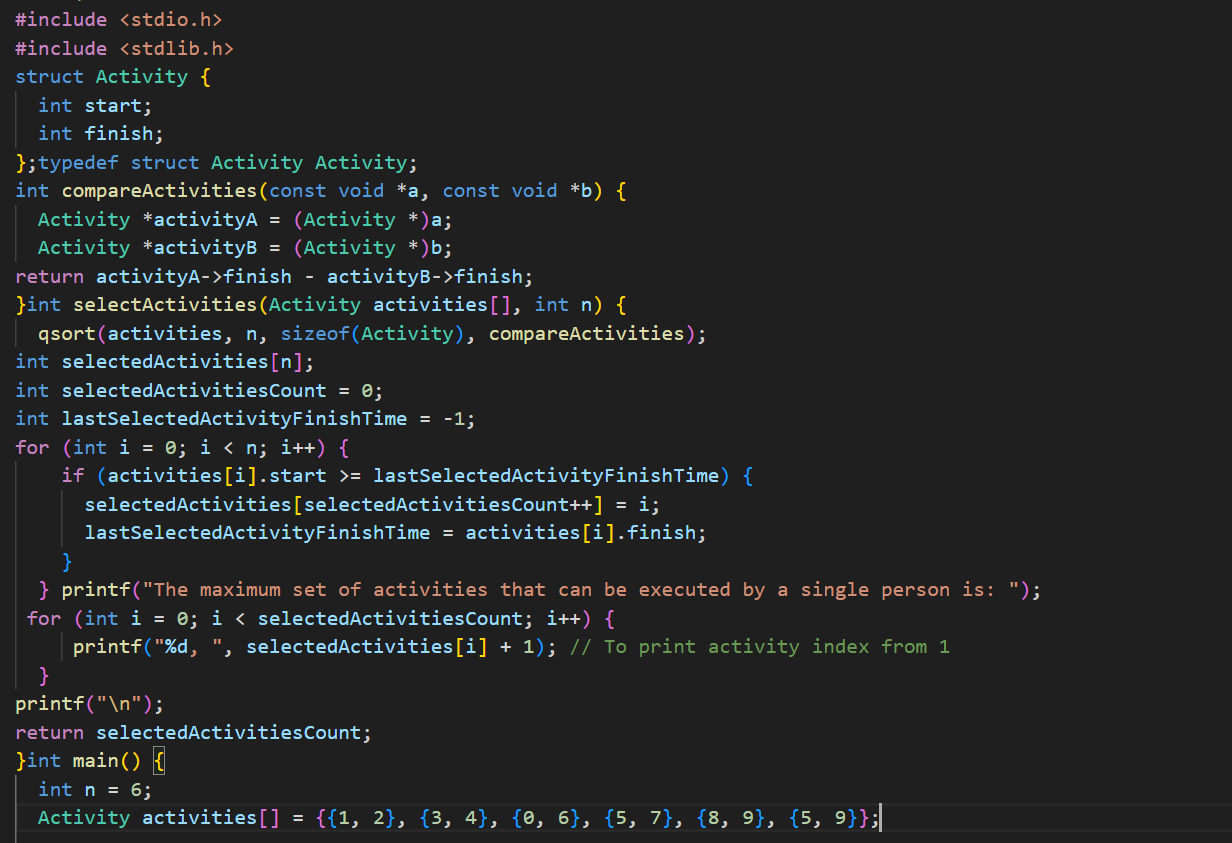
**EXPERIMENT-4: Title: GREEDY METHOD**

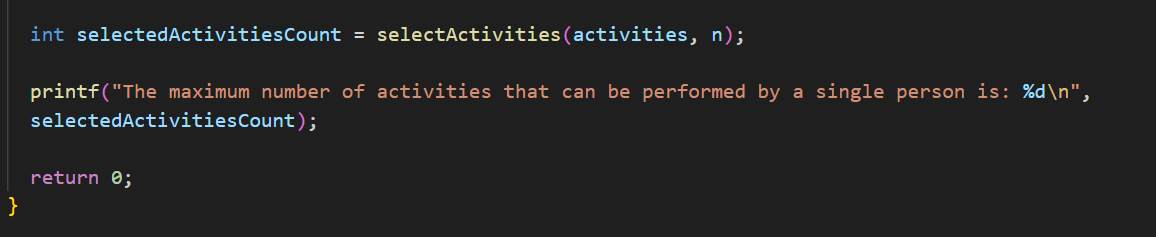
1. Implement the activity-selection problem (You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.

Example: Consider the following 6 activities.

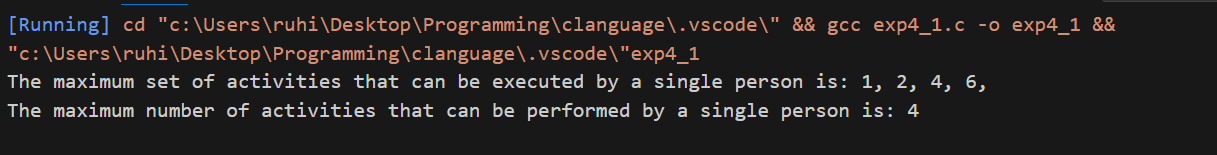
Start [] = {1, 3, 0, 5, 8, 5}; finish [] = {2, 4, 6, 7, 9, 9};

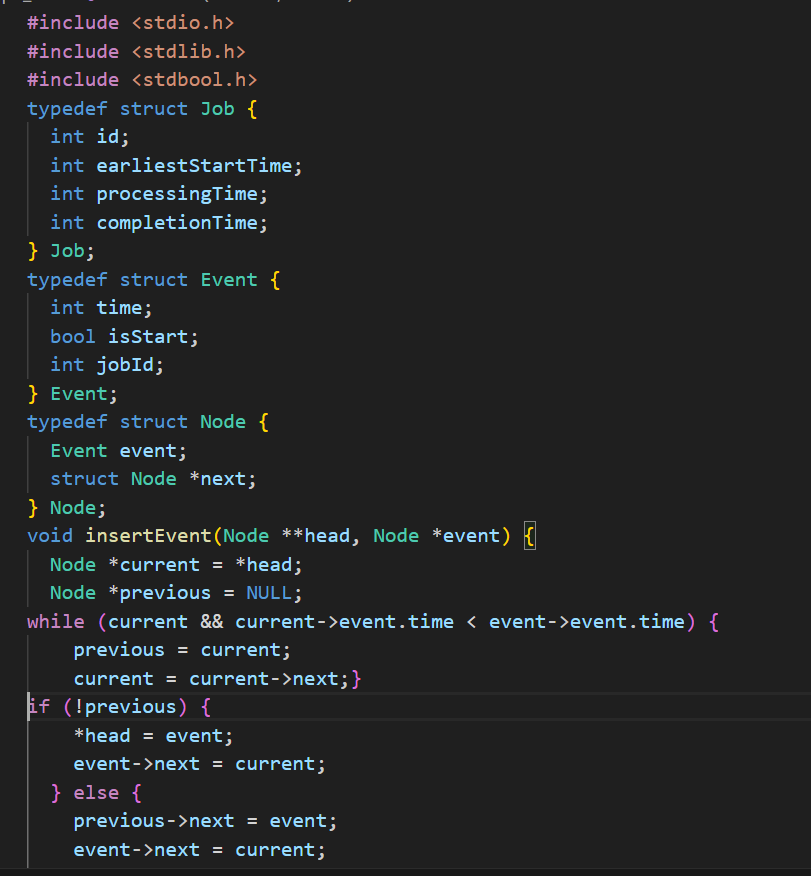
The maximum set of activities that can be executed by a single person is {0, 1, 3, 4}).

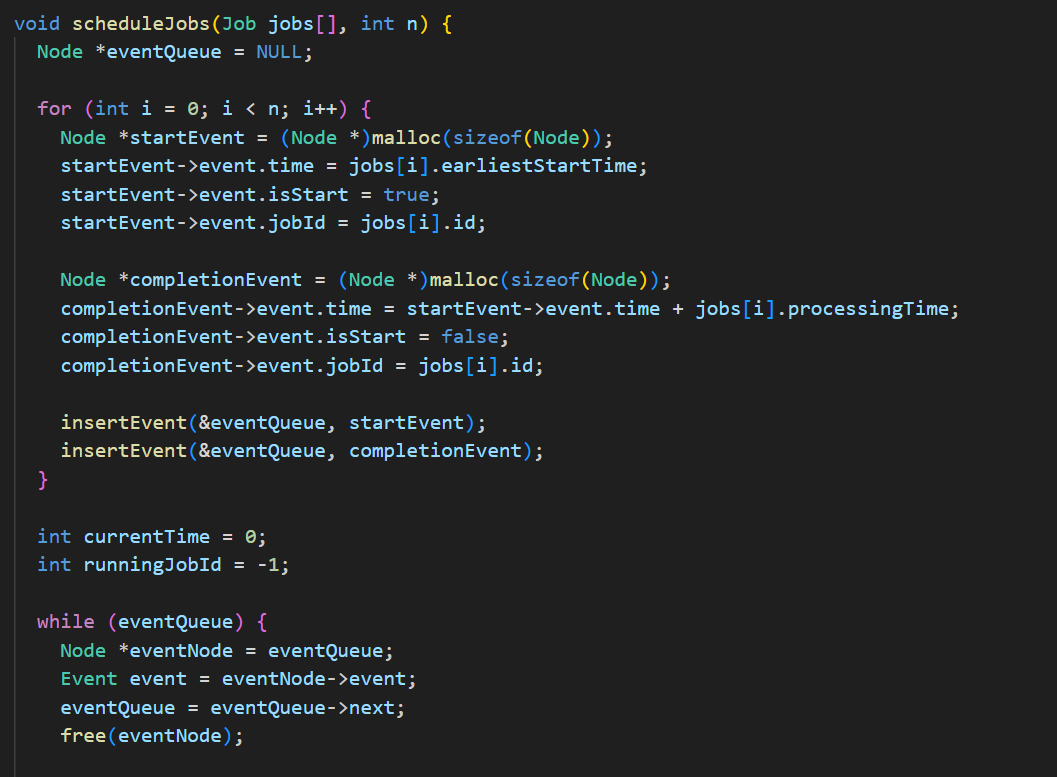


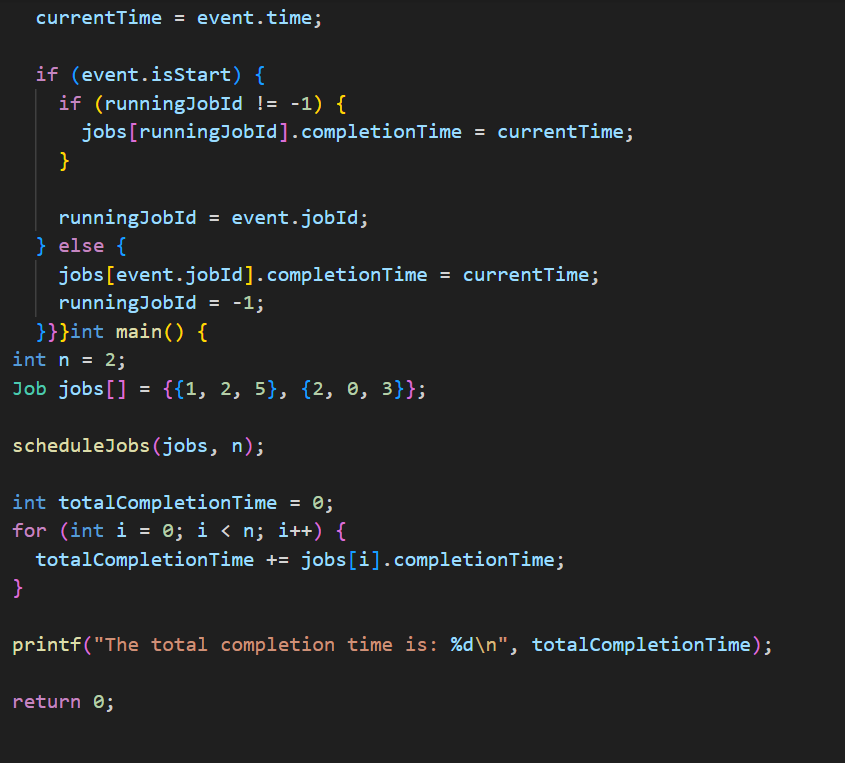


**Output:-**

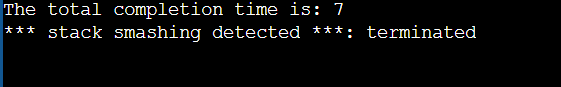
****2.Consider the following scheduling problem. You are given n jobs. Job i is specified by an earliest start time si, and a processing time pi. We consider a preemptive version of the problem where a job's execution can be suspended at any time and then completed later. For example if n = 2 and the input is s1 = 2, p1 = 5 and s2 = 0, p2 = 3, then a legal preemptive schedule is one in which job 2 runs from time 0 to 2 and is then suspended. Then job 1 runs from time 2 to 7 and secondly, job 2 is completed from time 7 to 8. The goal is to output a schedule that minimizes ΣCj = 1, where Cj is the time when job j is completed and j runs from 1 to n. In the example schedule given above, C1 =7 and C2=8.

****

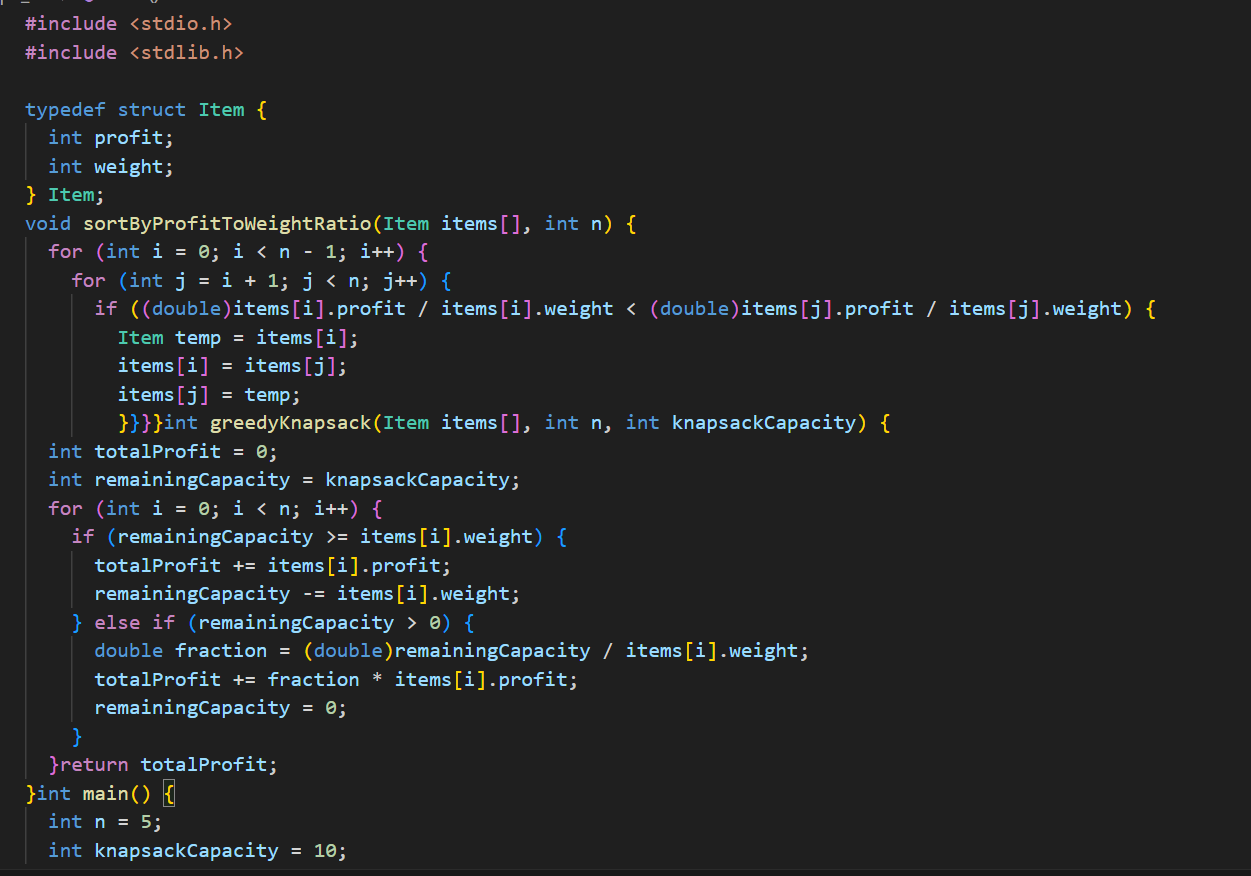
****

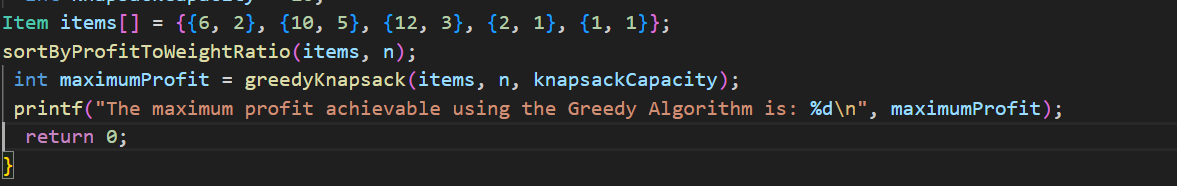
****

**Output:-**

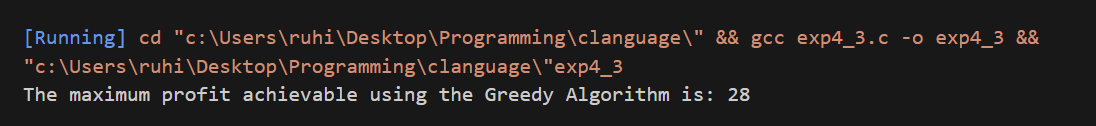
****

**3.**To find Optimal solution for a Knapsack Problem using Greedy Method

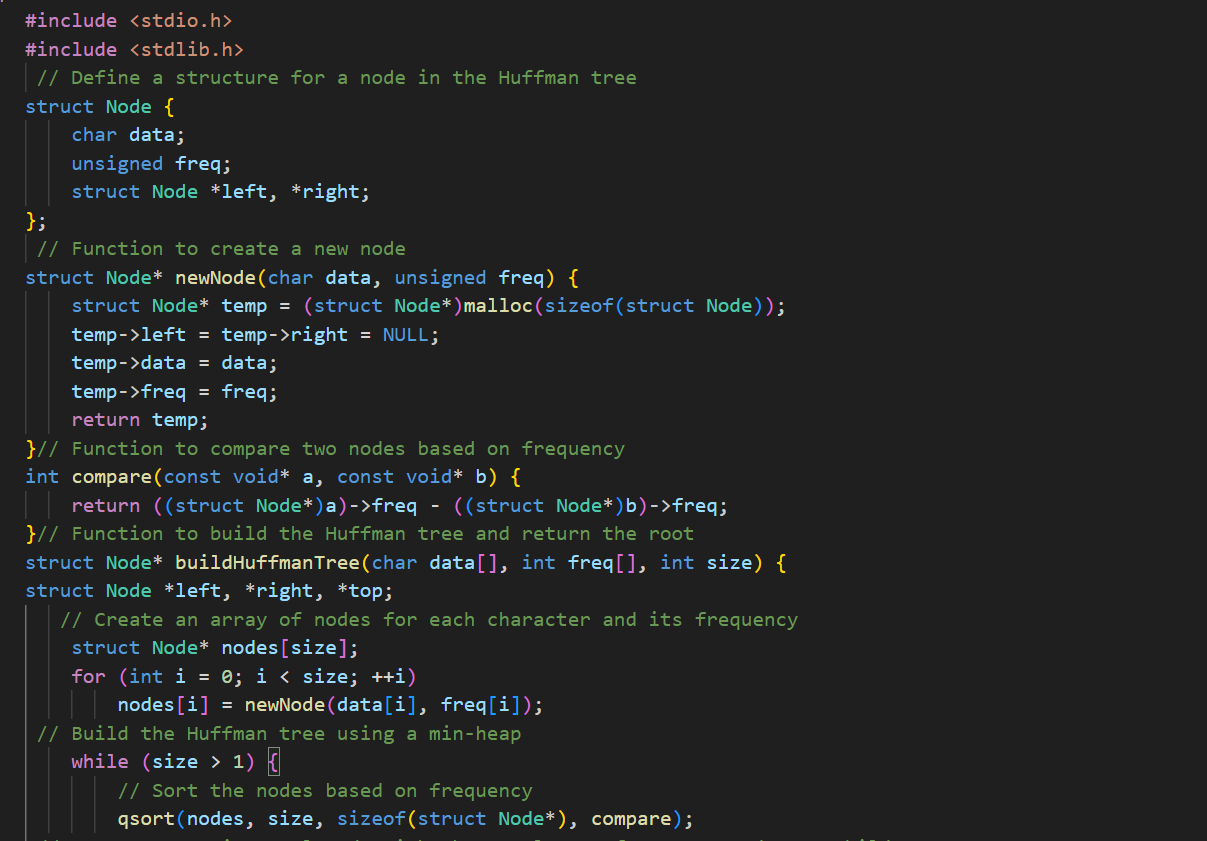


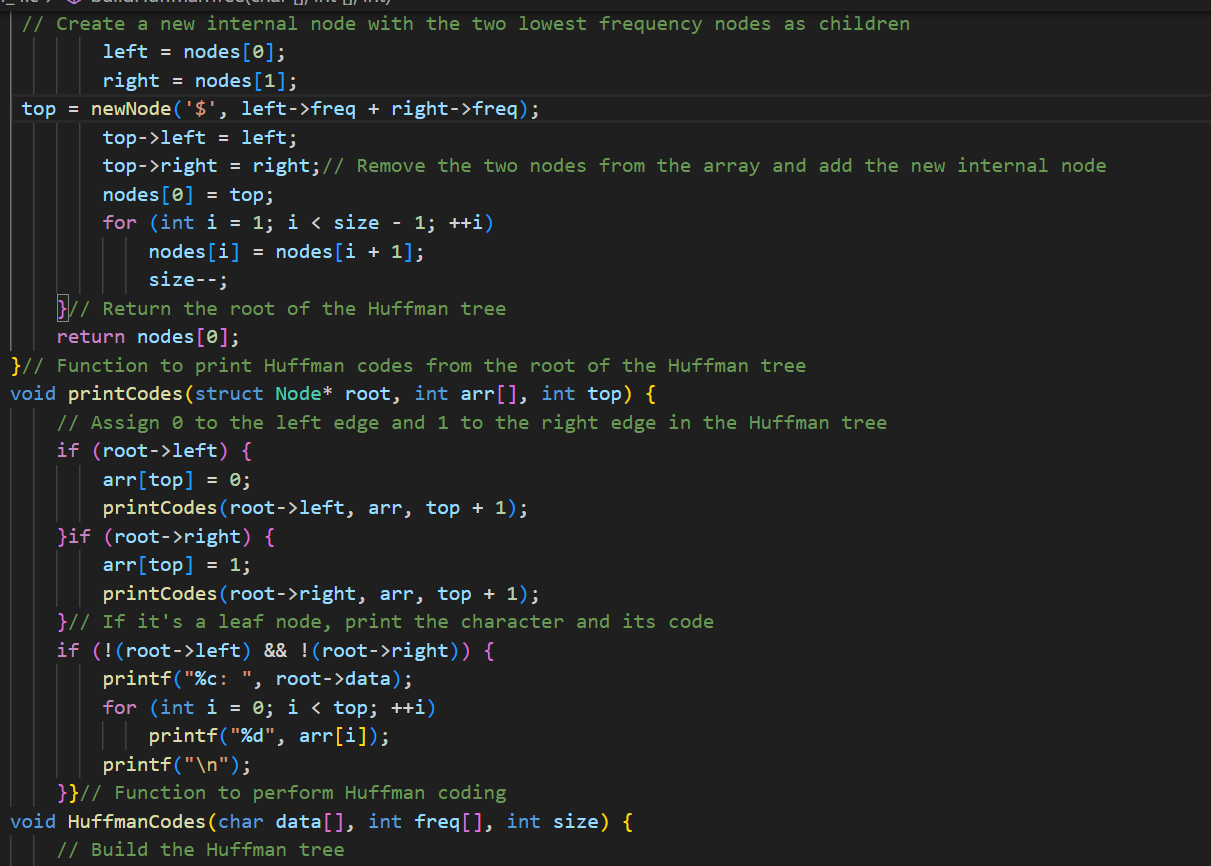


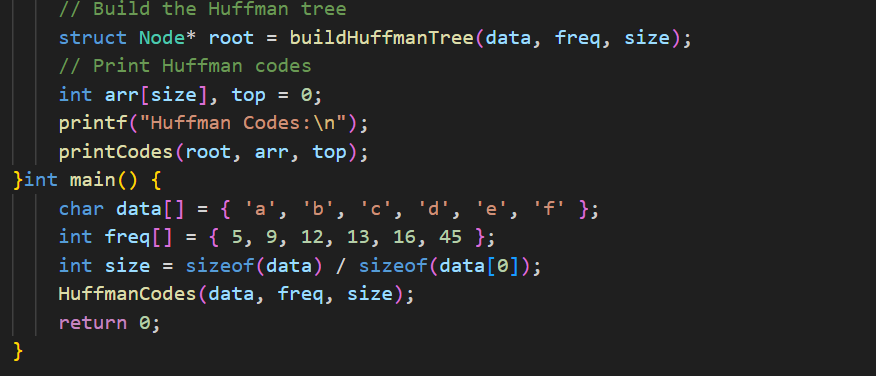
Output:-



**4.** Implement the file or code compression using Huffman’s algorithm.







Output:-

