DAA EXPERIENTIAL LEARNING ASSESSMENT

NAME: S.GANESH

ROLLNO: 23071A12F5

Q1) OPTIMAL CODE FOR QUICK SORT.

```
#include <iostream>
#include <algorithm>
using namespace std;

int medianOfThree(int arr[], int low, int high) {
   int mid = low + (high - low) / 2;
   if (arr[mid] < arr[low]) swap(arr[mid], arr[low]);
   if (arr[high] < arr[low]) swap(arr[high], arr[low]);
   if (arr[mid] < arr[high]) swap(arr[mid], arr[high]);
   return arr[high];
}

int partition(int arr[], int low, int high) {
   int pivot = medianOfThree(arr, low, high);
   int i = low - 1, j = high + 1;
   while (true) {
      do { i++; } while (arr[i] < pivot);
      do { j--; } while (arr[j] > pivot);
      if (i >= j) return j;
      swap(arr[i], arr[j]);
   }
}

void quickSort(int arr[], int low, int high) {
   while (low < high) {
      int p = partition(arr, low, high);
      if (p - low < high - p) {
         quickSort(arr, low, p);
```

```
            low = p + 1;  // Tail recursion optimization
        } else {
            quickSort(arr, p + 1, high);
            high = p;
        }
    }
}

int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;
    int arr[n];
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) cin >> arr[i];
    quickSort(arr, 0, n - 1);
    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
    return 0;
}
```

Output

```
Enter the number of elements: 8
Enter 8 elements: 34
7
23
32
5
62
32
4
Sorted array: 4 5 7 23 32 32 34 62


=== Code Execution Successful ===
```

2Q)OPTIMAL SEQUENCE OF JOB FOR MAXIMUM PROFIT.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Job {
   int id;     // Job ID
   int dead;   // Deadline
   int profit; // Profit
};

bool compare(Job a, Job b) {
   return a.profit > b.profit;
}

void jobSequencing(vector<Job> &jobs, int n) {
   // Step 1: Sort jobs in decreasing order of profit
   sort(jobs.begin(), jobs.end(), compare);

   int maxDeadline = 0;
   for (int i = 0; i < n; i++) {
      maxDeadline = max(maxDeadline, jobs[i].dead);
   }

   vector<int> slot(maxDeadline + 1, -1);
   int totalProfit = 0;
   vector<int> jobSequence;
```

```
    for (int i = 0; i < n; i++) {
        for (int j = jobs[i].dead; j > 0; j--) {
            if (slot[j] == -1) {
                slot[j] = jobs[i].id;
                jobSequence.push_back(jobs[i].id);
                totalProfit += jobs[i].profit;
                break;
            }
        }
    }

    cout << "Optimal job sequence for maximum profit: ";
    for (int job : jobSequence) {
        cout << job << " ";
    }
    cout << "\nTotal Profit: " << totalProfit << endl;
}

int main() {
    int n;
    cout << "Enter the number of jobs: ";
    cin >> n;
    vector<Job> jobs(n);
    cout << "Enter job details (ID, Deadline, Profit):\n";
    for (int i = 0; i < n; i++) {
        cin >> jobs[i].id >> jobs[i].dead >> jobs[i].profit;
    }
    jobSequencing(jobs, n);
    return 0;
}
```

Output

```
Enter the number of jobs: 6
Enter job details (ID, Deadline, Profit):
1
2
50
2
1
20
3
2
30
4
1
40
5
3
10
6
3
60
Optimal job sequence for maximum profit: 6 1 4
Total Profit: 150


=== Code Execution Successful ===
```