Assignment 6

A.) W(n): A function representing the maximum total weight that can be obtained from the set of n jobs.

B.) W(n) = max{ W(n-1), W(C(n)) + w_n } where C(n) is the index of the closest job j less than n such that job j is compatible with job n.

C.) The table is a dimensional array containing the maximum total weight that can be obtained from set of i jobs for any i^{th} index.

D.)

```
//Input: Job array, where each job object contains its start time, end time, and weight.
maxWeightJobSet(Job[] jobs){

    //Sort the jobs by their finish time. Complexity O(n*log(n))

    jobs.sort( j.finishTime );


    //Create Array of indices of closest job to any job i in said array;

    C = int[ jobs.length ];

    C[0] = 0;

    for(i = 1 to jobs.length){

        for(j = i-1 to 0){

            if(j == 0) C[i] = 0; break;

            if(jobs[i].startTime >= job[j].endTime){

                C[i] = j;

                break;

            }

        }

    }

    //Set up Result table

    weightTable = int[jobs.length];

    weightTable[0] = 0;
```

```
        //Fill Result Table
        for( i = 1 to jobs.length ){
            maxV = max(weightTable[i – 1], weightTable[C[i]] + jobs[i].weight);
            weightTable[i] = maxV;
        }
        return weightTable;
    }
```

E.)

```
    traceback(weightTable[], Job[] jobs){
        Stack<Job> jobSet;
        totalWeight = weightTable[weightTable.length - 1];
        for(i = weightTable.length-1 to 1)
            if(weightTable[i] > weightTable[i-1] && weightTable[i] == totalWeight)
                totalWeight -= jobs[i].weight;
                jobSet.add(jobs[i].id);
        }
        return jobSet;
    }
```

F.) $\sum_{i=1}^{n} 1 = n - 1 \rightarrow O(n)$