Experiment- 2.1

Student Name: Rishav Kumar **UID: 22MCC20039**

Branch: MCA **Section/Group:MCD-1/Grp-B**

Semester: 1st **Subject Code: 22CAP-646**

Subject Name: DAA LAB

1. Task to be done:

A. *Implement Fractional Knapsack problem using Greedy algorithm.*

B. *Implement 0/1 Knapsack problem using dynamic programming.*

2. Code for experiment/practical:

A:

```
#include <bits/stdc++.h>

using namespace std;

struct abc {

        int v, w;

        abc(int v, int w)

        {

                this->v = v;

                this->w = w;
        }
};
bool cmp(struct abc a, struct abc b)
{
        double r1 = (double)a.v / (double)a.w;
        double r2 = (double)b.v / (double)b.w;
        return r1 > r2;
}
double fractionalKnapsack(int W, struct abc arr[], int N)
{
        sort(arr, arr + N, cmp);
        double finalvalue = 0.0;
        for (int i = 0; i < N; i++) {
                if (arr[i].w <= W) {
                        W -= arr[i].w;
                        finalvalue += arr[i].v;
                }
                else {
                        finalvalue
                                += arr[i].v
                                * ((double)W / (double)arr[i].w);
                        break;
                }
        }
        return finalvalue;
}
int main()
```

```
{
        int W = 45;
        abc arr[] = { { 30, 20 }, { 50, 25 }, { 120, 31 } };


        int N = sizeof(arr) / sizeof(arr[0]);
        cout<<"Maximum value: ";
        cout <<fractionalKnapsack(W, arr, N);
        return 0;
}
```

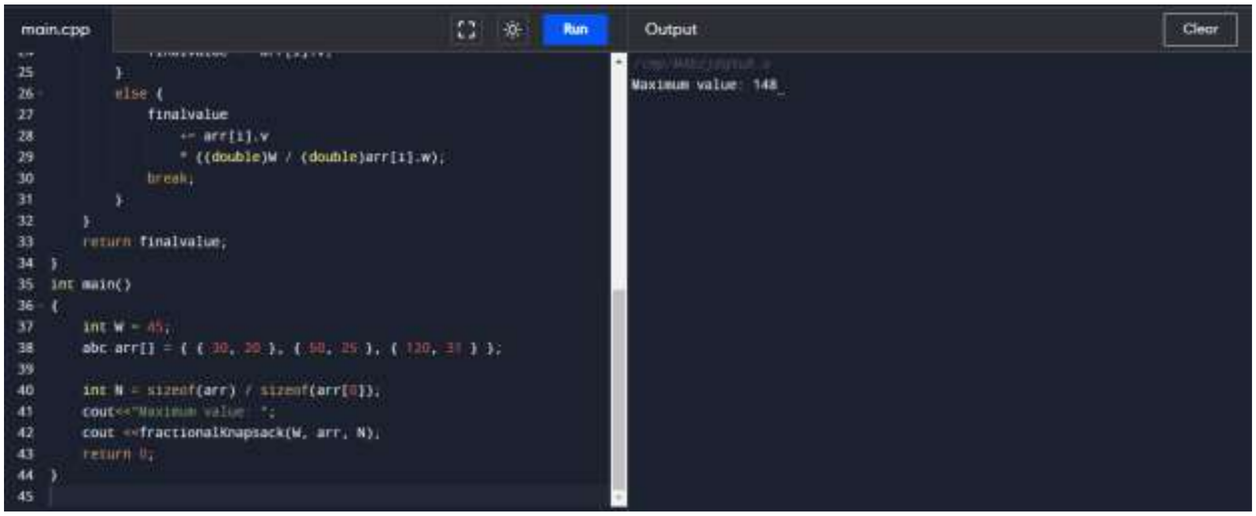
B:

```
#include <bits/stdc++.h>
using namespace std;
int max(int a, int b) { return (a > b) ? a : b; }
int knapSack(int W, int wt[], int val[], int n)
{
        if (n == 0 || W == 0)
                return 0;
        if (wt[n - 1] > W)
                return knapSack(W, wt, val, n - 1);
        else
                return max(
                        val[n - 1]
                                + knapSack(W - wt[n - 1],
                                        wt, val, n - 1),
                        knapSack(W, wt, val, n - 1));
}
```

```
int main()

{

        int val[] = { 70, 90, 110 };

        int wt[] = { 20, 30, 40 };

        int W = 60;

        int n = sizeof(val) / sizeof(val[0]);

        cout << knapSack(W, wt, val, n);

        return 0;

}
```

Output:

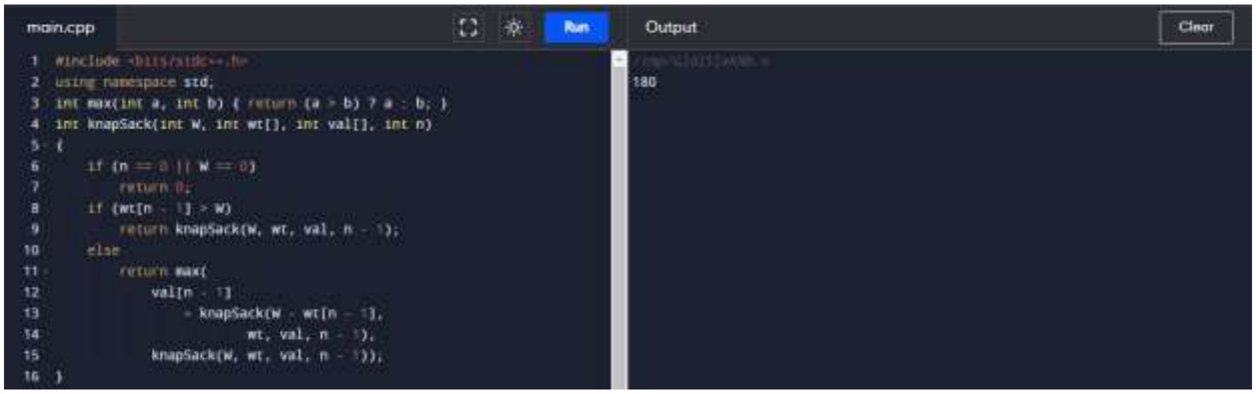
A:



```
        }
        else {
            finalvalue
                += arr[i].v
                * ((double)W / (double)arr[i].w);
            break;
        }
    }
    return finalvalue;
}
int main()
{
    int W = 45;
    abc arr[] = { { 30, 20 }, { 50, 25 }, { 120, 31 } };

    int N = sizeof(arr) / sizeof(arr[0]);
    cout<<"Maximum value: ";
    cout <<fractionalKnapsack(W, arr, N);
    return 0;
}
```

```
Maximum value: 148
```

B:



```
#include <bits/stdc++.h>
using namespace std;
int max(int a, int b) { return (a > b) ? a : b; }
int knapSack(int W, int wt[], int val[], int n)
{
    if (n == 0 || W == 0)
        return 0;
    if (wt[n - 1] > W)
        return knapSack(W, wt, val, n - 1);
    else
        return max(
            val[n - 1]
                + knapSack(W - wt[n - 1],
                        wt, val, n - 1),
            knapSack(W, wt, val, n - 1));
}
```

```
180
```