DAA Assignment 6

Name: Digvijay Pawar
Class: TY.Btech Comp B2
Gr.No: 21810344
Roll No: 322043

Branch and Bound Implementation (0/1 Knapsack Problem)

Code Implementation :

```
#include <bits/stdc++.h>
using namespace std;

struct Node
{
	int level, profit, bound;
	float weight;
};

struct Item
{
	float weight;
	int val;
};
```

```
bool cmp(Item a, Item b)
{
	double r1 = (double)a.val / a.weight;
	double r2 = (double)b.val / b.weight;
	return r1 > r2;
}

int bound(Node u, int n, int W, Item arr[])
{
	if (u.weight >= W)
		return 0;

	int profit_bound = u.profit;

	int j = u.level + 1;
	int totweight = u.weight;

	while ((j < n) && (totweight + arr[j].weight <= W))
	{
		totweight += arr[j].weight;
		profit_bound += arr[j].val;
		j++;
	}

	if (j < n)
		profit_bound += (W - totweight) * arr[j].val /
								arr[j].weight;
	return profit_bound;
}
```

```
int knapsack(int W, Item arr[], int n)
{
	sort(arr, arr + n, cmp);

	queue<Node> Q;
	Node u, v;

	u.level = -1;
	u.profit = u.weight = 0;
	Q.push(u);
	std::set<float> sol;
	int maxProfit = 0;
	while (!Q.empty())
	{
		u = Q.front();
		Q.pop();

		if (u.level == -1)
			v.level = 0;

		if (u.level == n-1)
			continue;

		v.level = u.level + 1;

		v.weight = u.weight + arr[v.level].weight;
		v.profit = u.profit + arr[v.level].val;

		if (v.weight <= W && v.profit > maxProfit)
			maxProfit = v.profit;
```

```
            v.bound = bound(v, n, W, arr);

            if (v.bound > maxProfit)
                Q.push(v);

            v.weight = u.weight;
            v.profit = u.profit;
            v.bound = bound(v, n, W, arr);
            if (v.bound > maxProfit)
            {
                Q.push(v);
                sol.insert(arr[v.level].weight);
            }
        }
  std::cout << "Element used for optimal solution: ";
  for (auto it = sol.begin(); it != sol.end(); it++)
       cout << *it << " ";
  std::cout<< '\n';
        return maxProfit;
}

int main()
{
        int W,n;
        std::cout << "Enter total entries: ";
        std::cin >> n;
        std::cout << "Enter Weight and Value" << '\n';
```

```
  Item arr[n];
  for (size_t i = 0; i < n; i++) {
    std::cin >> arr[i].weight>>arr[i].val;
  }
  std::cout << "Enter Total Capacity: ";
  std::cin >> W;
  int ans = knapsack(W, arr, n);
  cout << "Maximum possible profit = "<< ans<<endl;

  return 0;
}
```

Output: