Name	Aryaman Agarwal
UID no.	2021700002
Experiment No.	8

AIM:	To implement 0/1 Knapsack problem
PROBLEM STATEMENT :	To implement 0/1 Knapsack problem using branch and bound.
THEORY:	Branch and bound is an algorithmic technique used for solving optimization problems. It works by exploring the search space of a problem in a systematic way, dividing it into smaller sub-problems (branches) and using bounds to prune branches that cannot possibly lead to a better solution than the current best solution. The technique is widely used in combinatorial optimization problems, such as the 0/1 knapsack problem. The 0/1 knapsack problem is a classic optimization problem in which a knapsack has a limited capacity, and there are a number of items with different values and weights. The goal is to select a subset of the items that maximizes the total value while keeping the total weight within the capacity of the knapsack. This problem is called the 0/1 knapsack problem because each item can be either selected (1) or not selected (0). The Branch and Bound algorithm for the 0/1 knapsack problem works as follows: 1. Sort the items in decreasing order of their value-to-weight ratios. 2. Create a root node of the search tree with no items selected and set the current profit to 0. 3. Calculate the upper bound of the profit for the root node using the bound function. 4. Add the root node to a priority queue, sorted by the upper bound of the profit. 5. While the priority queue is not empty: a. Remove the node with the highest priority (i.e., the one with the highest upper bound of the profit) from the queue. b. If the node represents a complete solution, update the current best solution if the profit is higher than the current best solution. c. If the node represents an incomplete solution, create two

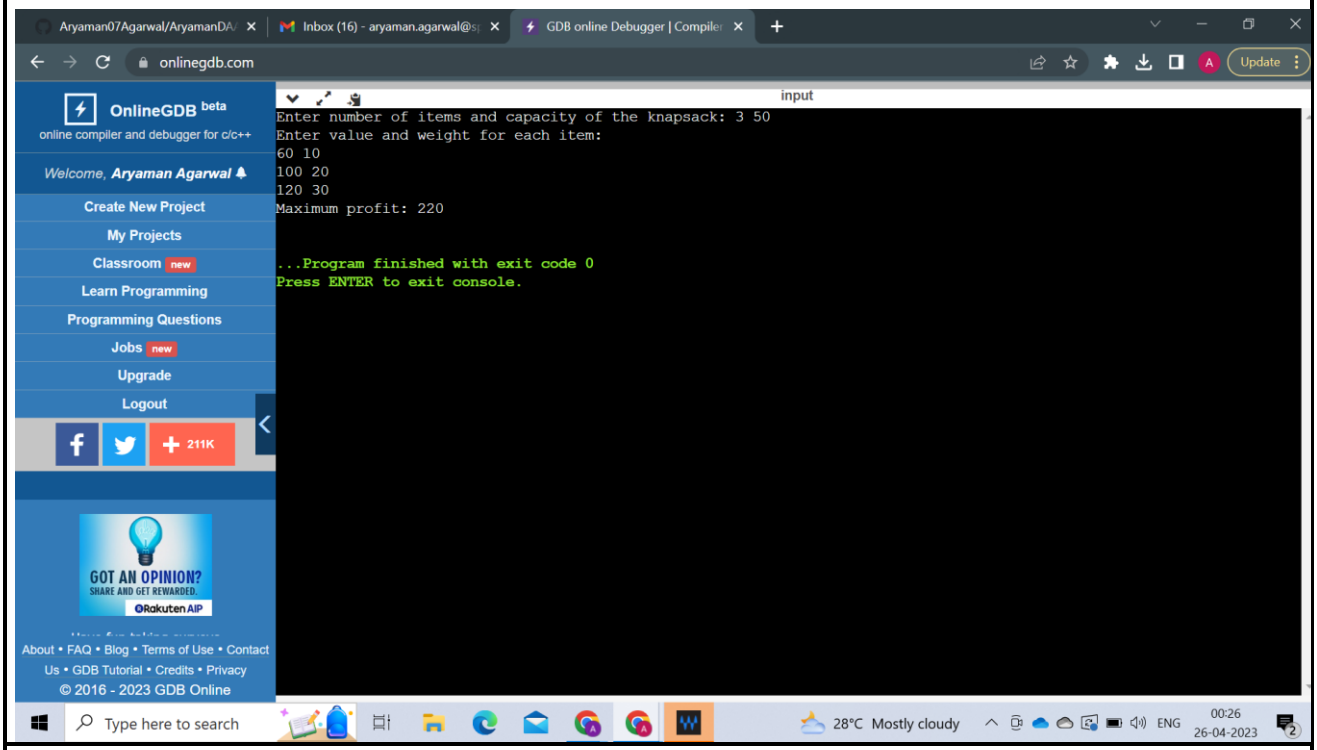
	child nodes by including or excluding the next item in the knapsack. Calculate the upper bounds of the profits for the child nodes and add them to the priority queue. d. Prune child nodes that cannot possibly lead to a better solution than the current best solution. The bound function is used to calculate the upper bound of the profit for a given node in the search tree. It calculates the maximum profit that can be obtained by adding fractions of items to the knapsack, based on the remaining capacity of the knapsack and the remaining items. The bound function is used to prune child nodes that cannot possibly lead to a better solution than the current best solution. Overall, Branch and Bound algorithm is a powerful technique for solving optimization problems. In the case of the 0/1 knapsack problem, it can efficiently find the optimal solution by exploring only a subset of the search space.

Program:

```
 #include <stdio.h>
#include <stdlib.h>

#define MAX_N 100
#define MAX_W 1000

typedef struct item {
    int value;
    int weight;
    float bound;
} Item;

int max_profit = 0;
int n, W;
Item items[MAX_N];

void read_input() {
    printf("Enter number of items and capacity of the knapsack: ");
    scanf("%d %d", &n, &W);
    printf("Enter value and weight for each item:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &items[i].value, &items[i].weight);
        items[i].bound = (float)items[i].value / (float)items[i].weight;
    }
}

int bound(int i, int w, int profit) {
    if (w > W) {
        return 0;
    }
    int b = profit;
    int j = i + 1;
    int weight = w;
    while ((j < n) && (weight + items[j].weight <= W)) {
        b += items[j].value;
        weight += items[j].weight;
```

```
        j++;
    }
    if (j < n) {
        b += (int)((float)(W - weight) * items[j].bound);
    }
    return b;
}

void branch_and_bound(int i, int w, int profit) {
    if (w > W) {
        return;
    }
    if (i == n) {
        if (profit > max_profit) {
            max_profit = profit;
        }
        return;
    }
    if (bound(i, w, profit) <= max_profit) {
        return;
    }
    branch_and_bound(i + 1, w, profit);
    branch_and_bound(i + 1, w + items[i].weight, profit + items[i].value);
}

int main() {
    read_input();
    branch_and_bound(0, 0, 0);
    printf("Maximum profit: %d\n", max_profit);
    return 0;
}
```

RESULT:



CONCLUSION: I understood 0/1 Knapsack problem and implemented it using Branch and Bound in C.