```
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>

using namespace std;

struct Item {
    int weight;
    int value;
};

struct Node {
    int level;
    int value;
    int weight;
    double bound;

    bool operator<(const Node& other) const {
        return bound < other.bound;
    }
};

double bound(Node u, int n, int capacity, vector<Item>& items) {
    if (u.weight >= capacity) return 0.0;

    double boundValue = u.value;
    int j = u.level + 1;
    int totalWeight = u.weight;

    while (j < n && totalWeight + items[j].weight <= capacity) {
        totalWeight += items[j].weight;
        boundValue += items[j].value;
```

```
        j++;
    }

    if (j < n) {
        boundValue += (capacity - totalWeight) *
static_cast<double>(items[j].value) / items[j].weight;
    }

    return boundValue;
}

int knapsackBranchAndBound(int capacity, vector<Item>& items) {
    int n = items.size();
    sort(items.begin(), items.end(), [](const Item& a, const Item& b)
{
        return a.value * b.weight > b.value * a.weight;
    });

    priority_queue<Node> pq;
    Node u, v;
    u.level = -1;
    u.value = u.weight = 0;
    u.bound = bound(u, n, capacity, items);
    int maxVal = 0;

    pq.push(u);

    while (!pq.empty()) {
        u = pq.top();
        pq.pop();

        if (u.bound > maxVal) {
            v.level = u.level + 1;
            v.weight = u.weight + items[v.level].weight;
```

```
            v.value = u.value + items[v.level].value;

            if (v.weight <= capacity && v.value > maxVal) {
                maxVal = v.value;
            }

            v.bound = bound(v, n, capacity, items);

            if (v.bound > maxVal) {
                pq.push(v);
            }

            v.weight = u.weight;
            v.value = u.value;
            v.bound = bound(v, n, capacity, items);

            if (v.bound > maxVal) {
                pq.push(v);
            }
        }
    }

    return maxVal;
}

int main() {
    int n, capacity;
    cout << "Enter the number of items: ";
    cin >> n;

    vector<Item> items(n);

    cout << "Enter the weight and value of each item:" << endl;
```

```
    for (int i = 0; i < n; ++i) {
        cin >> items[i].weight >> items[i].value;
    }

    cout << "Enter the maximum capacity of the knapsack: ";
    cin >> capacity;

    int maxValue = knapsackBranchAndBound(capacity, items);
    cout << "Maximum value in the knapsack: " << maxValue << endl;

    return 0;
}


//OUTPUT:
Enter the number of items: 3
Enter the weight and value of each item:
10 60
20 100
30 120
Enter the maximum capacity of the knapsack: 50
Maximum value in the knapsack: 220
```