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

// Structure for an item which stores weight and
// corresponding value of Item
struct Item {
	int profit, weight;

	// Constructor
	Item(int profit, int weight)
	{
		this->profit = profit;
		this->weight = weight;
	}
};

// Comparison function to sort Item
// according to profit/weight ratio
static bool cmp(struct Item a, struct Item b)
{
	double r1 = (double)a.profit / (double)a.weight;
	double r2 = (double)b.profit / (double)b.weight;
	return r1 > r2;
}

// Main greedy function to solve problem
double fractionalKnapsack(int W, struct Item arr[], int N)
{
	// Sorting Item on basis of ratio
	sort(arr, arr + N, cmp);

	double finalvalue = 0.0;

	// Looping through all items
	for (int i = 0; i < N; i++) {

		// If adding Item won't overflow,
		// add it completely
		if (arr[i].weight <= W) {
			W -= arr[i].weight;
			finalvalue += arr[i].profit;
		}
```

```
            // If we can't add current Item,
            // add fractional part of it
            else {
                finalvalue
                    += arr[i].profit
                    * ((double)W / (double)arr[i].weight);
                break;
            }
        }

        // Returning final value
        return finalvalue;
}

// Driver code
int main()
{
        int W = 50;
        Item arr[] = { { 60, 10 }, { 100, 20 }, { 120, 30 } };
        int N = sizeof(arr) / sizeof(arr[0]);

        // Function call
        cout << fractionalKnapsack(W, arr, N);
        return 0;
}
```

Output

240