ROLL NO:- 33252

```
#include <iostream>
#include <vector>
#include<algorithm>
using namespace std;

void heapify(vector<vector<int>>& arr, int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left][1]/arr[left][0] > arr[largest][1]/arr[largest][0])
        largest = left;

    if (right < n && arr[right][1]/arr[right][0] > arr[largest][1]/arr[largest][0])
        largest = right;

    if (largest != i) {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}

void heapSort(vector<vector<int>>& arr, int n) {

    for (int i = n / 2 - 1; i >= 0; i--)
```

```
            heapify(arr, n, i);

    for (int i = n - 1; i > 0; i--) {
            swap(arr[0], arr[i]);
            heapify(arr, i, 0);
    }
}

bool comp(vector<int> &a,vector<int> &b){
    if( (a[1]/a[0]) >= (b[1]/b[0])){
            return true;
    }
    return false;
}

int f(vector<vector<int>>& arr, int n, int w) {
    heapSort(arr, n);
    reverse(arr.begin(),arr.end());
    //sort(arr.begin(),arr.end(),comp);
    int tot_val = 0;
    for (int i = 0; i < n; i++) {
            if (arr[i][0] <= w) {
            tot_val += arr[i][1];
            w = w - arr[i][0];
            } else {
            tot_val += (arr[i][1] / arr[i][0]) * w;
```

```
            break;
            }
    }

    return tot_val;
}

int main() {
    int t;
    cout << "Enter the number of test cases: ";
    cin >> t;

    while (t--) {
            int n;
            cout << "Enter the number of items: ";
            cin >> n;

            vector<vector<int>> arr(n, vector<int>(2));
            cout << "Enter weight and value for each item:" << endl;
            for (int i = 0; i < n; i++) {
            cin >> arr[i][0] >> arr[i][1];
            }

            int w;
            cout << "Enter the weight capacity of the knapsack: ";
            cin >> w;
```

```
        double max_value = f(arr, n, w);
        cout << "Maximum value that can be obtained = " << max_value << endl;
    }

    return 0;
}
```

OUTPUT

```
1
3
10 60
20 100
30 120
50
```

Enter the weight capacity of the knapsack: Maximum value that can be obtained = 240