

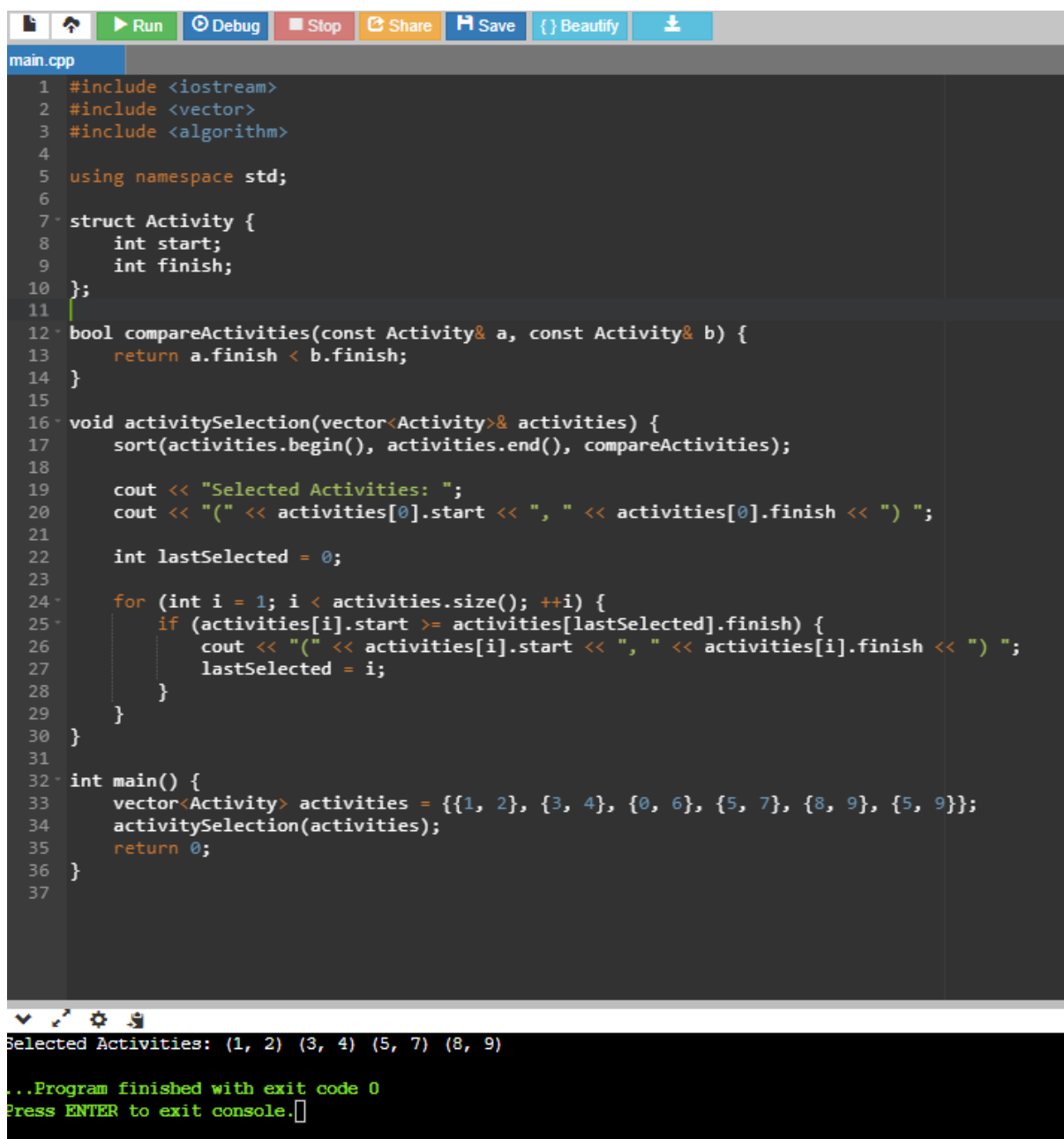
# DAA LAB ASSIGNMENT 2

NAME->Chandranshu Bhardwaj

ROLL NO.->102203797

Q1.) Activity selection?

Ans->



The image shows a screenshot of a C++ program in a code editor. The editor has a toolbar at the top with buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The file name is 'main.cpp'. The code is as follows:

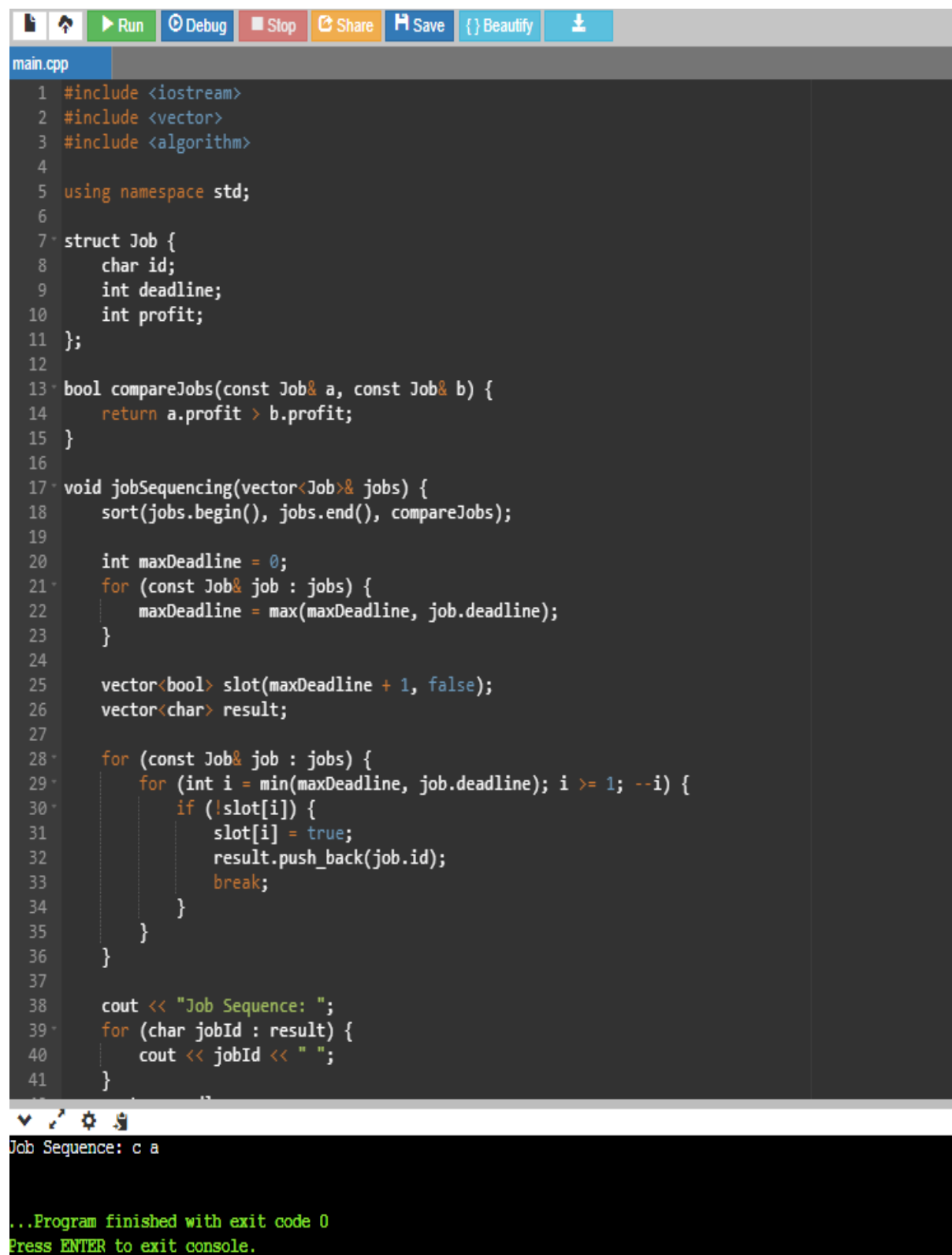
```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 using namespace std;
6
7 struct Activity {
8     int start;
9     int finish;
10 };
11
12 bool compareActivities(const Activity& a, const Activity& b) {
13     return a.finish < b.finish;
14 }
15
16 void activitySelection(vector<Activity>& activities) {
17     sort(activities.begin(), activities.end(), compareActivities);
18
19     cout << "Selected Activities: ";
20     cout << "(" << activities[0].start << ", " << activities[0].finish << ") ";
21
22     int lastSelected = 0;
23
24     for (int i = 1; i < activities.size(); ++i) {
25         if (activities[i].start >= activities[lastSelected].finish) {
26             cout << "(" << activities[i].start << ", " << activities[i].finish << ") ";
27             lastSelected = i;
28         }
29     }
30 }
31
32 int main() {
33     vector<Activity> activities = {{1, 2}, {3, 4}, {0, 6}, {5, 7}, {8, 9}, {5, 9}};
34     activitySelection(activities);
35     return 0;
36 }
37
```

Below the code editor, the output of the program is shown in a console window:

```
Selected Activities: (1, 2) (3, 4) (5, 7) (8, 9)
...Program finished with exit code 0
Press ENTER to exit console.
```

Q2.) Job Sequence?

Ans->



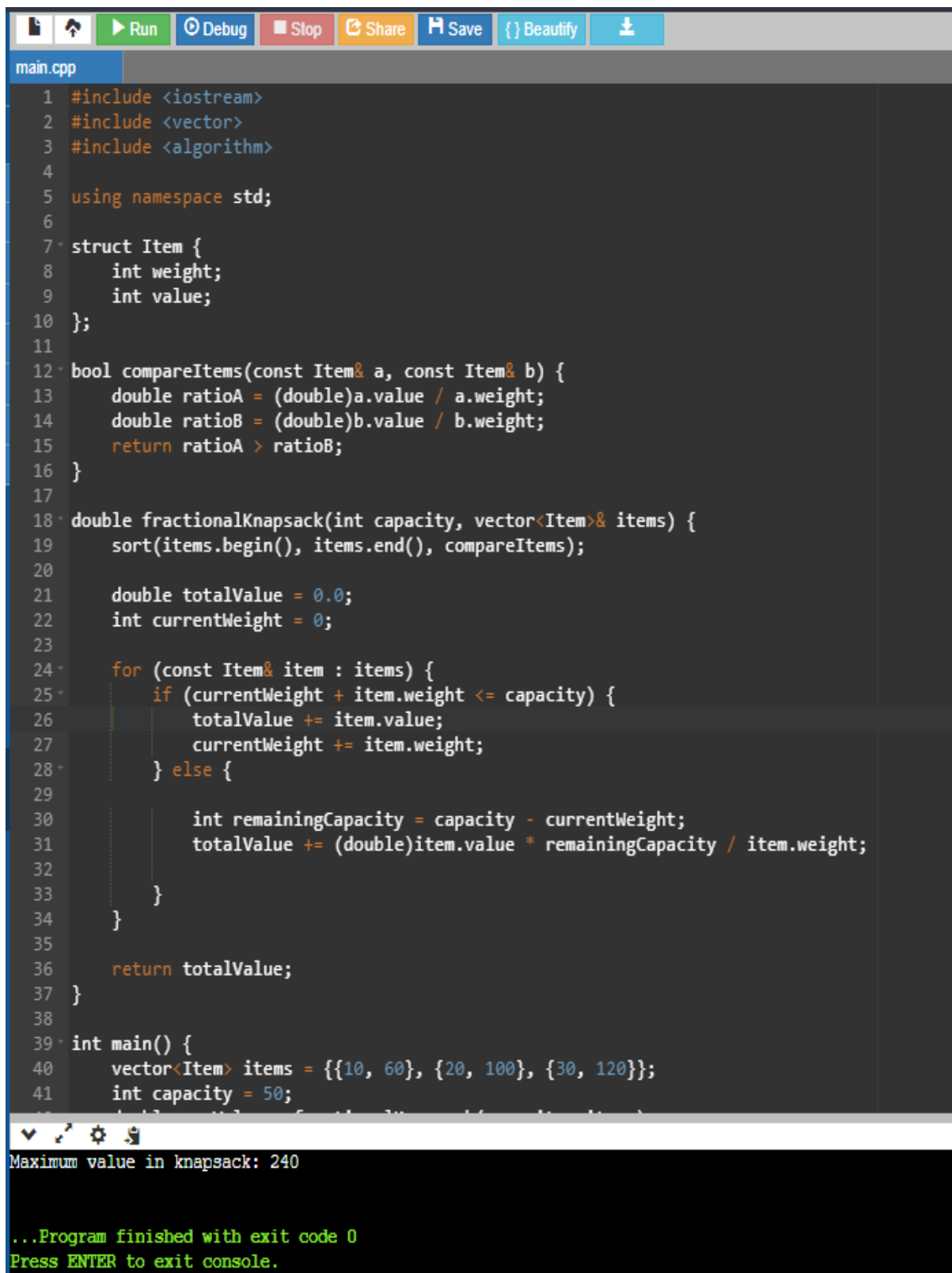
```
main.cpp
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  struct Job {
8      char id;
9      int deadline;
10     int profit;
11 };
12
13 bool compareJobs(const Job& a, const Job& b) {
14     return a.profit > b.profit;
15 }
16
17 void jobSequencing(vector<Job>& jobs) {
18     sort(jobs.begin(), jobs.end(), compareJobs);
19
20     int maxDeadline = 0;
21     for (const Job& job : jobs) {
22         maxDeadline = max(maxDeadline, job.deadline);
23     }
24
25     vector<bool> slot(maxDeadline + 1, false);
26     vector<char> result;
27
28     for (const Job& job : jobs) {
29         for (int i = min(maxDeadline, job.deadline); i >= 1; --i) {
30             if (!slot[i]) {
31                 slot[i] = true;
32                 result.push_back(job.id);
33                 break;
34             }
35         }
36     }
37
38     cout << "Job Sequence: ";
39     for (char jobId : result) {
40         cout << jobId << " ";
41     }
42 }
```

Job Sequence: c a

...Program finished with exit code 0  
Press ENTER to exit console.

Q3.)Fractional knapsack?

Ans->



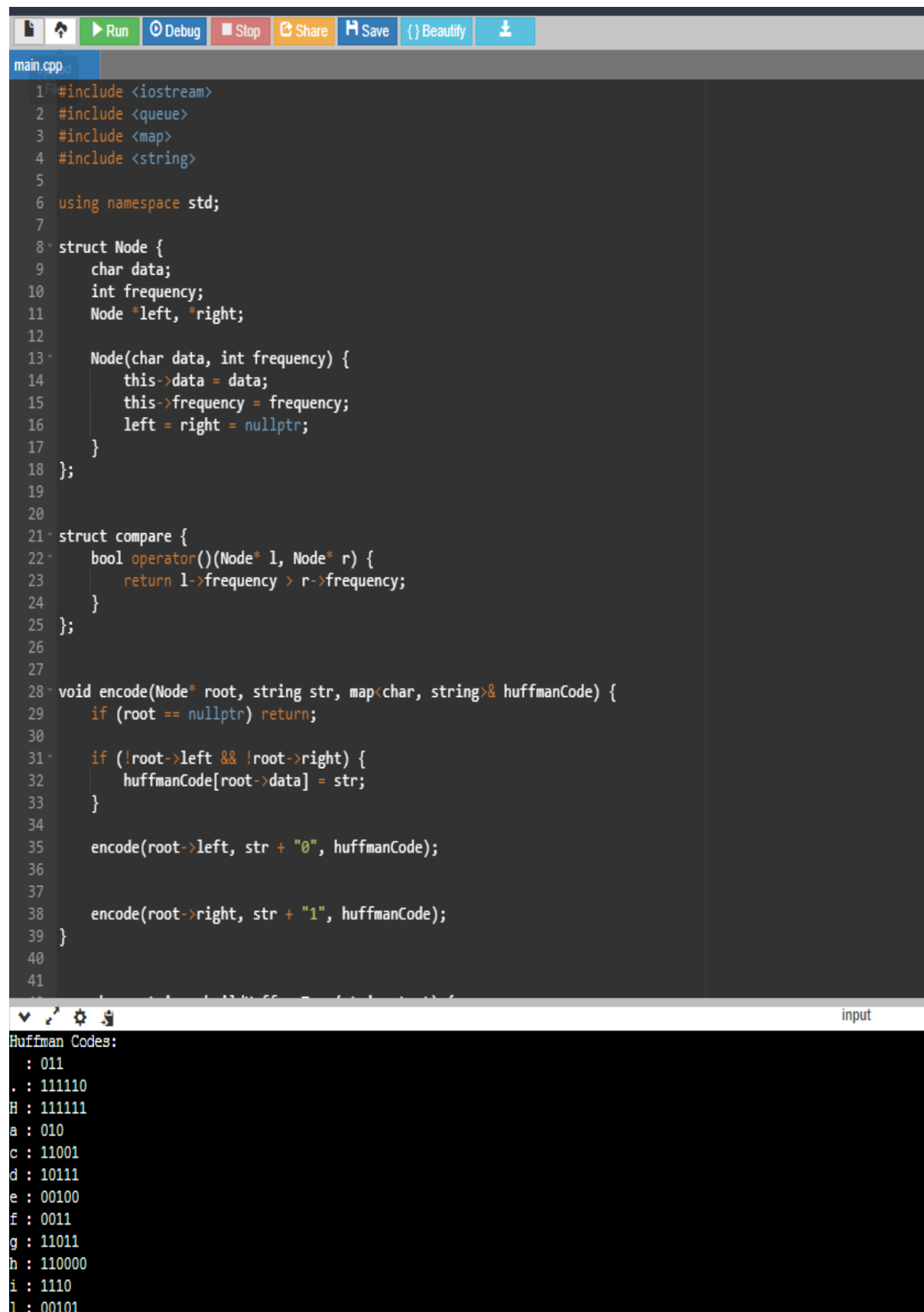
```
main.cpp
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  struct Item {
8      int weight;
9      int value;
10 };
11
12 bool compareItems(const Item& a, const Item& b) {
13     double ratioA = (double)a.value / a.weight;
14     double ratioB = (double)b.value / b.weight;
15     return ratioA > ratioB;
16 }
17
18 double fractionalKnapsack(int capacity, vector<Item>& items) {
19     sort(items.begin(), items.end(), compareItems);
20
21     double totalValue = 0.0;
22     int currentWeight = 0;
23
24     for (const Item& item : items) {
25         if (currentWeight + item.weight <= capacity) {
26             totalValue += item.value;
27             currentWeight += item.weight;
28         } else {
29
30             int remainingCapacity = capacity - currentWeight;
31             totalValue += (double)item.value * remainingCapacity / item.weight;
32         }
33     }
34
35     return totalValue;
36 }
37
38
39 int main() {
40     vector<Item> items = {{10, 60}, {20, 100}, {30, 120}};
41     int capacity = 50;
42
43     cout << "Maximum value in knapsack: " << fractionalKnapsack(capacity, items) << endl;
44
45     return 0;
46 }
```

Maximum value in knapsack: 240

...Program finished with exit code 0  
Press ENTER to exit console.

### Q3.) Huffman Coding?

Ans->



The image shows a C++ IDE with a code editor and a terminal window. The code editor displays a C++ program for Huffman coding. The program includes headers for `<iostream>`, `<queue>`, `<map>`, and `<string>`. It uses the `std` namespace and defines a `Node` struct with `data`, `frequency`, and pointers to `left` and `right` nodes. A `compare` struct is used to compare nodes based on frequency. The `encode` function recursively builds the Huffman tree and generates the codes. The terminal window shows the output of the program, displaying the Huffman codes for each character.

```
1 #include <iostream>
2 #include <queue>
3 #include <map>
4 #include <string>
5
6 using namespace std;
7
8 struct Node {
9     char data;
10    int frequency;
11    Node *left, *right;
12
13    Node(char data, int frequency) {
14        this->data = data;
15        this->frequency = frequency;
16        left = right = nullptr;
17    }
18 };
19
20
21 struct compare {
22     bool operator()(Node* l, Node* r) {
23         return l->frequency > r->frequency;
24     }
25 };
26
27
28 void encode(Node* root, string str, map<char, string>& huffmanCode) {
29     if (root == nullptr) return;
30
31     if (!root->left && !root->right) {
32         huffmanCode[root->data] = str;
33     }
34
35     encode(root->left, str + "0", huffmanCode);
36
37     encode(root->right, str + "1", huffmanCode);
38 }
39
40
41
```

Huffman Codes:

```
 : 011
. : 111110
H : 111111
a : 010
c : 11001
d : 10111
e : 00100
f : 0011
g : 11011
h : 110000
i : 1110
l : 00101
```

Run

Debug

Stop

Share

Save

Beautify

main.cpp

```
41
42 map<char, string> buildHuffmanTree(string text) {
43
44     map<char, int> freq;
45     for (char c : text) {
46         freq[c]++;
47     }
48
49
50     priority_queue<Node*, vector<Node*>, compare> pq;
51     for (auto pair : freq) {
52         pq.push(new Node(pair.first, pair.second));
53     }
54
55     while (pq.size() != 1) {
56         Node* left = pq.top(); pq.pop();
57         Node* right = pq.top(); pq.pop();
58
59         Node* newNode = new Node('$', left->frequency + right->frequency);
60         newNode->left = left;
61         newNode->right = right;
62
63         pq.push(newNode);
64     }
65
66     Node* root = pq.top();
67     map<char, string> huffmanCode;
68     encode(root, "", huffmanCode);
69
70     return huffmanCode;
71 }
72
73 int main() {
74     string text = "Huffman coding is a data compression algorithm.";
75     map<char, string> huffmanCode = buildHuffmanTree(text);
76
77     cout << "Huffman Codes:\n";
78     for (auto pair : huffmanCode) {
79         cout << pair.first << " : " << pair.second << endl;
80     }
81
82 }
```

input

```
i : 1110
l : 00101
m : 1001
n : 1000
o : 000
p : 10110
r : 11010
s : 1010
t : 11110
u : 110001
```