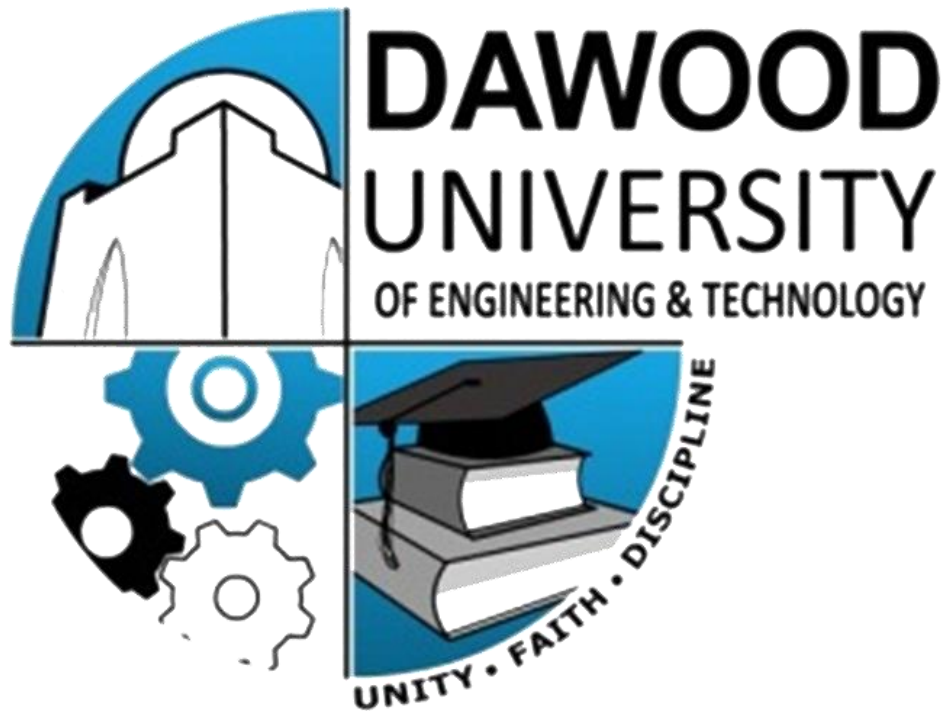


# OEL#2



Name: Muhammad Ilyas khan

Roll No: 23-CS-115

Subject: DSA

Q1) A simple text editor needs to implement an Undo feature. Every time a user performs an operation (like typing or deleting text), it should be possible to undo the last action. Use a stack to achieve this functionality.

```

1  #include <iostream>
2  #include <stack>
3  #include <string>
4
5  using namespace std;
6
7  class TextEditor {
8  private:
9      string text;
10     stack<string> history;
11
12 public:
13     void addText(const string& newText) {
14         history.push(text);
15         text += newText;
16         cout << "Added text: \"" << newText << "\"\n";
17     }
18
19     void deleteText(int count) {
20         if (count > text.size()) {
21             cout << "Cannot delete more characters than present.\n";
22             return;
23         }
24         history.push(text);
25         text.erase(text.size() - count);
26         cout << "Deleted last " << count << " characters\n";
27     }
28
29     void undo() {
30         if (!history.empty()) {
31             text = history.top();
32             history.pop();
33             cout << "Undo performed. Current text: \"" << text << "\"\n";
34         } else {
35             cout << "No operations to undo.\n";
36         }
37     }
38
39     void displayText() const {
40         cout << "Current text: \"" << text << "\"\n";
41     }
42 };
43
44 int main() {
45     TextEditor editor;
46
47     editor.addText("Hello");
48     editor.displayText();
49
50     editor.addText(" World");
51     editor.displayText();
52
53     editor.deleteText(5);
54     editor.displayText();
55
56     editor.undo();
57     editor.displayText();
58
59     editor.undo();
60     editor.displayText();
61
62     editor.undo();
63     editor.displayText();
64
65     return 0;
66 }
67

```

OUTPUT:

```
Added text: "Hello"
Current text: "Hello"
Added text: " World"
Current text: "Hello World"
Deleted last 5 characters
Current text: "Hello "
Undo performed. Current text: "Hello World"
Current text: "Hello World"
Undo performed. Current text: "Hello"
Current text: "Hello"
Undo performed. Current text: ""
Current text: ""
[1] + Done                                "/usr/bin/gdb" --interpreter=mi --tty=s
```

Q2) Check the CHILDREN SUM PROPERTY in the Binary tree, i.e. for every node data value must be equal to the sum of data values of the left and right child.

```

1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int data;
6      Node* left;
7      Node* right;
8
9      Node(int val) {
10         data = val;
11         left = right = nullptr;
12     }
13 };
14
15 bool checkChildrenSumProperty(Node* root) {
16     if (root == nullptr || (root->left == nullptr && root->right == nullptr)) {
17         return true;
18     }
19
20     int leftData = (root->left) ? root->left->data : 0;
21     int rightData = (root->right) ? root->right->data : 0;
22
23     if (root->data == leftData + rightData &&
24         checkChildrenSumProperty(root->left) &&
25         checkChildrenSumProperty(root->right)) {
26         return true;
27     }
28
29     return false;
30 }
31
32 int main() {
33     Node* root = new Node(10);
34     root->left = new Node(8);
35     root->right = new Node(2);
36     root->left->left = new Node(3);
37     root->left->right = new Node(5);
38     root->right->left = new Node(2);
39
40     if (checkChildrenSumProperty(root)) {
41         cout << "The tree satisfies the Children Sum Property.\n";
42     } else {
43         cout << "The tree does not satisfy the Children Sum Property.\n";

```

```

15 bool checkChildrenSumProperty(Node* root) {
16     if (root == nullptr || (root->left == nullptr && root->right == nullptr)) {
17         return true;
18     }
19
20     int leftData = (root->left) ? root->left->data : 0;
21     int rightData = (root->right) ? root->right->data : 0;
22
23     if (root->data == leftData + rightData &&
24         checkChildrenSumProperty(root->left) &&
25         checkChildrenSumProperty(root->right)) {
26         return true;
27     }
28
29     return false;
30 }
31
32 int main() {
33     Node* root = new Node(10);
34     root->left = new Node(8);
35     root->right = new Node(2);
36     root->left->left = new Node(3);
37     root->left->right = new Node(5);
38     root->right->left = new Node(2);
39
40     if (checkChildrenSumProperty(root)) {
41         cout << "The tree satisfies the Children Sum Property.\n";
42     } else {
43         cout << "The tree does not satisfy the Children Sum Property.\n";
44     }
45
46     return 0;
47 }
48

```

OUTPUT;

```

The tree satisfies the Children Sum Property.
[1] + Done          "/usr/bin/gdb" --interpreter=mi --tty=$
muhammad.ilvas.khan@muhammad.ilvas.khan-Precision-3541:~/Desktop/files/u

```