

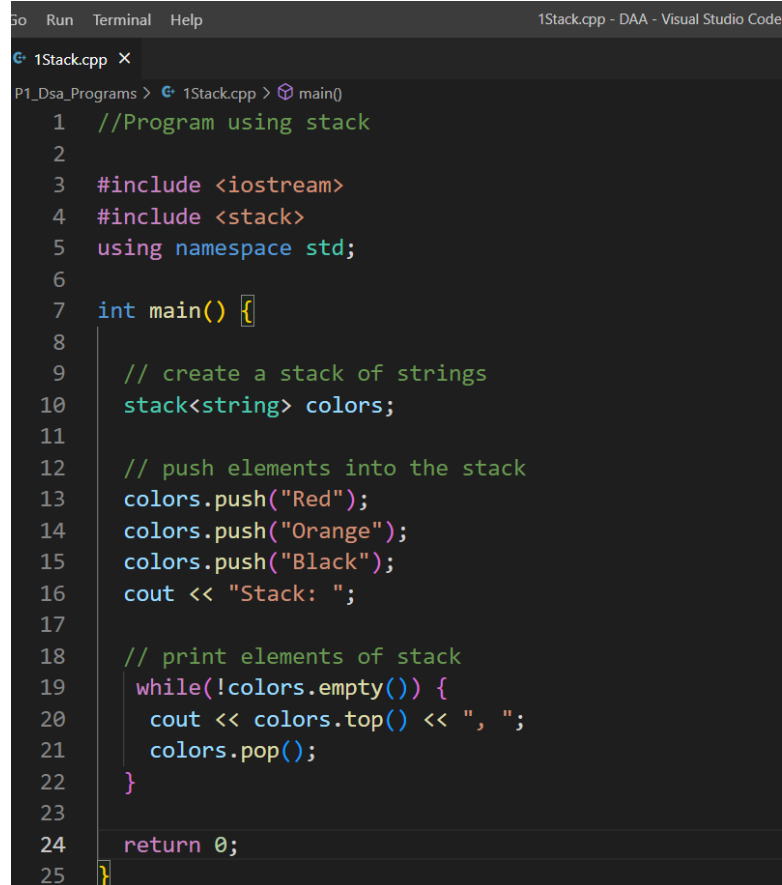
DAA Practical No 1

Aim: Write C/C++ code to implement concept of

- 1)Stack
- 2)Queue
- 3)Linked List
- 4)Trees
- 5) Graphs

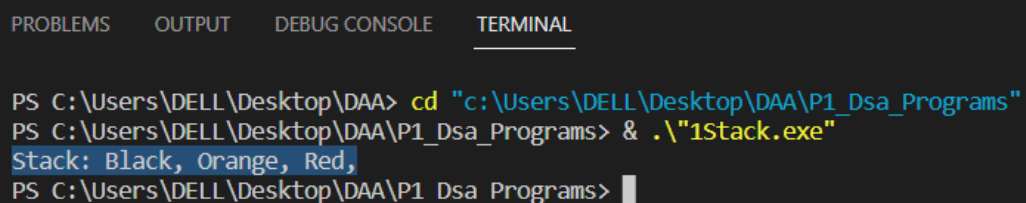
1)Stack

--> Program:



```
Go Run Terminal Help 1Stack.cpp - DAA - Visual Studio Code
1Stack.cpp x
P1_Dsa_Programs > 1Stack.cpp > main()
1 //Program using stack
2
3 #include <iostream>
4 #include <stack>
5 using namespace std;
6
7 int main() {
8
9     // create a stack of strings
10    stack<string> colors;
11
12    // push elements into the stack
13    colors.push("Red");
14    colors.push("Orange");
15    colors.push("Black");
16    cout << "Stack: ";
17
18    // print elements of stack
19    while(!colors.empty()) {
20        cout << colors.top() << ", ";
21        colors.pop();
22    }
23
24    return 0;
25 }
```

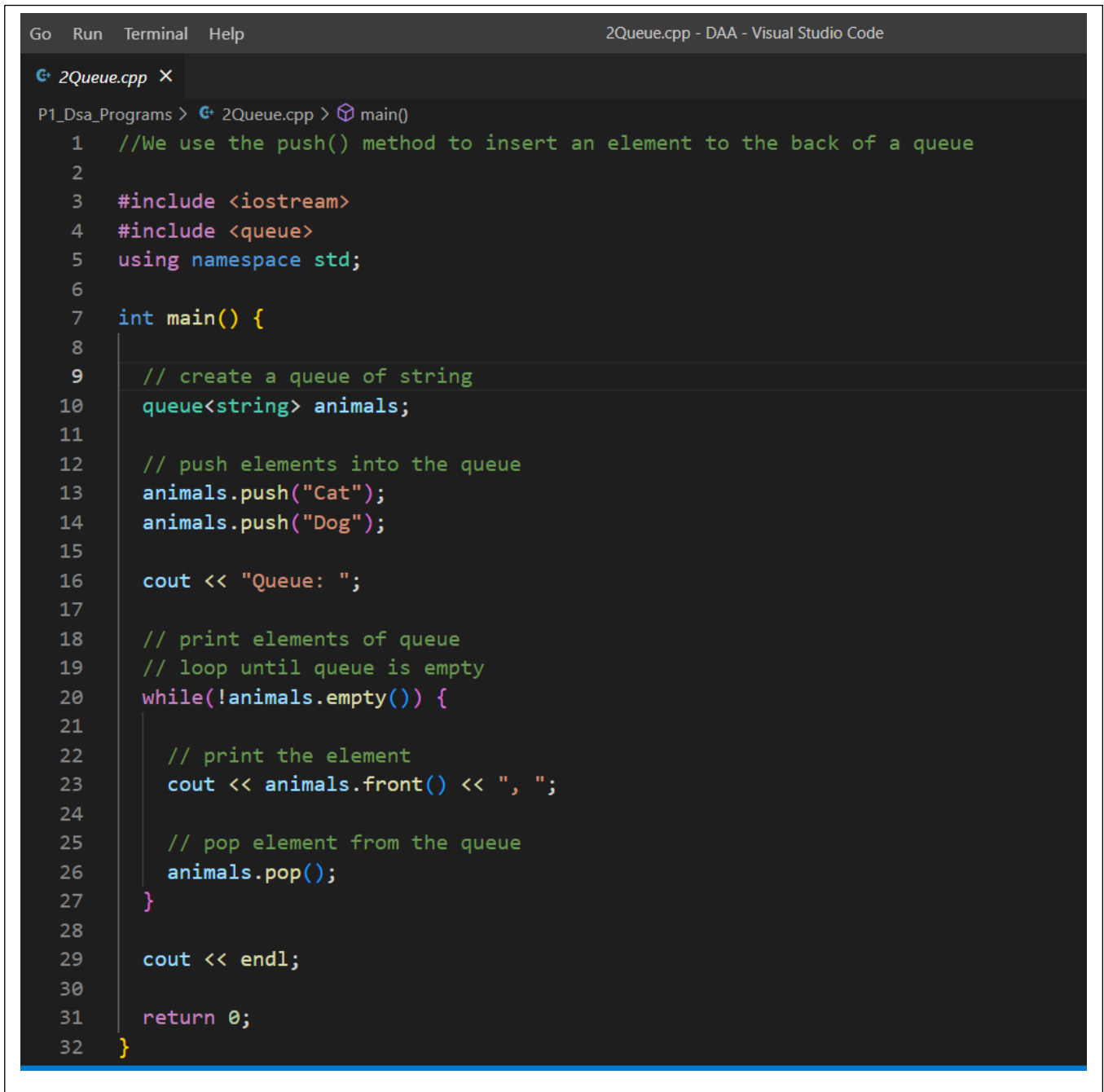
Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\DELL\Desktop\DAA> cd "c:\Users\DELL\Desktop\DAA\P1_Dsa_Programs"
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> & .\"1Stack.exe"
Stack: Black, Orange, Red,
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> |
```

2)Queue

-- >Program:

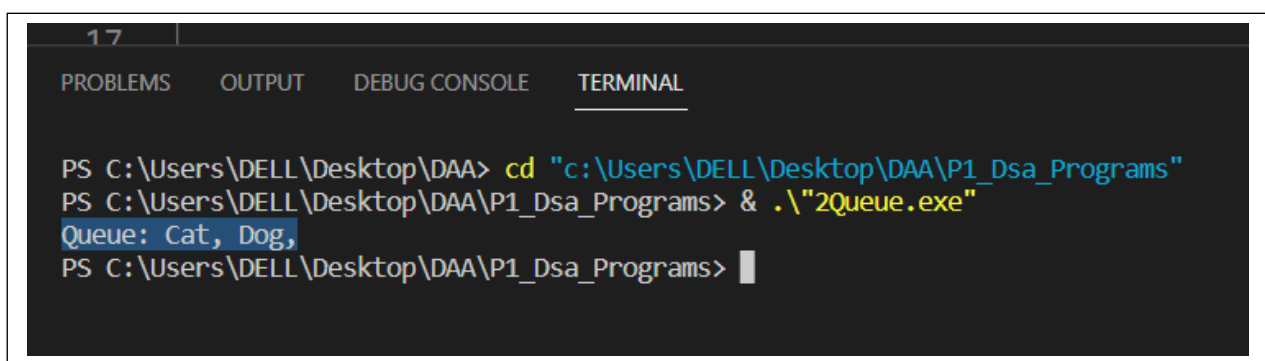


The screenshot shows the Visual Studio Code editor with a file named 2Queue.cpp open. The code is written in C++ and implements a queue using the STL queue container. The code includes necessary headers, uses the std namespace, and defines a main function that creates a queue of strings, pushes 'Cat' and 'Dog', and then prints them in a loop until the queue is empty.

```
Go Run Terminal Help 2Queue.cpp - DAA - Visual Studio Code

2Queue.cpp X
P1_Dsa_Programs > 2Queue.cpp > main()
1 //We use the push() method to insert an element to the back of a queue
2
3 #include <iostream>
4 #include <queue>
5 using namespace std;
6
7 int main() {
8
9     // create a queue of string
10    queue<string> animals;
11
12    // push elements into the queue
13    animals.push("Cat");
14    animals.push("Dog");
15
16    cout << "Queue: ";
17
18    // print elements of queue
19    // loop until queue is empty
20    while(!animals.empty()) {
21
22        // print the element
23        cout << animals.front() << ", ";
24
25        // pop element from the queue
26        animals.pop();
27    }
28
29    cout << endl;
30
31    return 0;
32 }
```

Output:



The screenshot shows the Windows Command Prompt with the following commands and output:

```
17
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\DELL\Desktop\DAA> cd "c:\Users\DELL\Desktop\DAA\P1_Dsa_Programs"
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> & .\"2Queue.exe"
Queue: Cat, Dog,
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> █
```

3)Linked List

-->Program:

```
Go Run Terminal Help 3linkedlist.cpp - DAA - V

3linkedlist.cpp X

P1_Dsa_Programs > 3linkedlist.cpp > main()
1 // Linked list implementation in C++
2
3 #include <bits/stdc++.h>
4 #include <iostream>
5 using namespace std;
6
7 // Creating a node
8 class Node {
9     public:
10     int value;
11     Node* next;
12 };
13
14 int main() {
15     Node* head;
16     Node* one = NULL;
17     Node* two = NULL;
18     Node* three = NULL;
19
20     // allocate 3 nodes in the heap
21     one = new Node();
22     two = new Node();
23     three = new Node();
24
25     // Assign value values
26     one->value = 1;
27     two->value = 2;
28     three->value = 3;
29
30     // Connect nodes
31     one->next = two;
32     two->next = three;
33     three->next = NULL;
34
35     // print the linked list value
36     head = one;
37     while (head != NULL) {
38         cout << head->value;
39         head = head->next;
40     }
41 }
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\DELL\Desktop\DAA> cd "c:\Users\DELL\Desktop\DAA\P1_Dsa_Programs"
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> & .\"3linkedlist.exe"
123
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> |
```

4)Trees

-->Program:

```
Go Run Terminal Help 4tree.cpp - DAA - Visual Studio Code

4tree.cpp X
P1_Dsa_Programs > 4tree.cpp > main()
1 // Binary Tree in C++
2
3 #include <stdlib.h>
4
5 #include <iostream>
6
7 using namespace std;
8
9 struct node {
10     int data;
11     struct node *left;
12     struct node *right;
13 };
14
15 // New node creation
16 struct node *newNode(int data) {
17     struct node *node = (struct node *)malloc(sizeof(struct node));
18
19     node->data = data;
20
21     node->left = NULL;
22     node->right = NULL;
23     return (node);
24 }
25
26 // Traverse Preorder
27 void traversePreOrder(struct node *temp) {
28     if (temp != NULL) {
29         cout << " " << temp->data;
30         traversePreOrder(temp->left);
31         traversePreOrder(temp->right);
32     }
33 }
34
35 // Traverse Inorder
36 void traverseInOrder(struct node *temp) {
37     if (temp != NULL) {
38         traverseInOrder(temp->left);
39         cout << " " << temp->data;
40         traverseInOrder(temp->right);
41     }
42 }
43
44 // Traverse Postorder
45 void traversePostOrder(struct node *temp) {
46     if (temp != NULL) {
47         traversePostOrder(temp->left);
48         traversePostOrder(temp->right);
49         cout << " " << temp->data;
50     }
51 }
52
```

```

53 int main() {
54     struct node *root = newNode(1);
55     root->left = newNode(2);
56     root->right = newNode(3);
57     root->left->left = newNode(4);
58
59     cout << "preorder traversal: ";
60     traversePreOrder(root);
61     cout << "\nInorder traversal: ";
62     traverseInOrder(root);
63     cout << "\nPostorder traversal: ";
64     traversePostOrder(root);
65 }

```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS C:\Users\DELL\Desktop\DAA> cd "c:\Users\DELL\Desktop\DAA\P1_Dsa_Programs"
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs> & .\"4tree.exe"
preorder traversal:  1 2 4 3
Inorder traversal:  4 2 1 3
Postorder traversal: 4 2 3 1
PS C:\Users\DELL\Desktop\DAA\P1_Dsa_Programs>

```

5)Graphs

-->Program:

Go Run Terminal Help 5graphs.cpp - DAA - Visual Studio Code

5graphs.cpp X

P1_Dsa_Programs > 5graphs.cpp > addEdge(int, int)

```

1 // C++ program to print DFS traversal from a given vertex in a given graph
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 // Graph class represents a directed graph using adjacency list representation
7 class Graph {
8 public:
9     map<int, bool> visited;
10    map<int, list<int> > adj;
11
12    // function to add an edge to graph
13    void addEdge(int v, int w);
14
15    // DFS traversal of the vertices reachable from v
16    void DFS(int v);
17 };
18 void Graph::addEdge(int v, int w)
19 {
20     adj[v].push_back(w); // Add w to v's list.
21 }
22 void Graph::DFS(int v)
23 {
24     // Mark the current node as visited and print it
25     visited[v] = true;
26     cout << v << " ";
27
28     // Recur for all the vertices adjacent to this vertex
29     list<int>::iterator i;
30     for (i = adj[v].begin(); i != adj[v].end(); ++i)
31         if (!visited[*i])
32             DFS(*i);
33 }
34

```

```

35 int main()
36 {
37     // Create a graph given in the above diagram
38     Graph g;
39     g.addEdge(0, 1);
40     g.addEdge(0, 2);
41     g.addEdge(1, 2);
42     g.addEdge(2, 0);
43     g.addEdge(2, 3);
44     g.addEdge(3, 3);
45
46     cout << "Following is Depth First Traversal"
47           " (starting from vertex 2) \n";
48
49     // Function call
50     g.DFS(2);
51
52     return 0;
53 }
54

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\DELL\Desktop\DAA> cd "c:\Users\DELL\Desktop\DAA\P2_Dsa_Programs"
PS C:\Users\DELL\Desktop\DAA\P2_Dsa_Programs> & .\"1StackUsingLinkedList.exe"
44 -> 33 -> 22 -> 11
Top element is 44
22 -> 11
Top element is 22
PS C:\Users\DELL\Desktop\DAA\P2_Dsa_Programs>

```