

```
1 /
   *****
2
3 Welcome to GDB Online.
4 GDB online is an online compiler and debugger tool for C, C++, Python,
   PHP, Ruby,
5 C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML,
   CSS, JS
6 Code, Compile, Run and Debug online from anywhere in world.
7
8 *****
   *****/
9 /*
10 Concatenate two input singly linked lists l1 and l2 into a new output
   linked list
11 that contains all the nodes of
12 l1 followed by all the nodes of l2.
13 */
14
15 #include <iostream>
16 #include <string>
17
18 // Forward declaration
19 template <typename T> class SLinkedList;
20 template <typename T> void concat(SLinkedList<T>& l1, SLinkedList<T>& l2,
21 SLinkedList<T>& lout);
22
23 template <typename T>
24 class SNode {
25 private:
26 T elem;
27 SNode<T>* next;
28 friend class SLinkedList<T>; // Added "class" keyword for proper friend
   declaration
29 friend void concat<T>(SLinkedList<T>& l1, SLinkedList<T>& l2,
   SLinkedList<T>&
30 lout);
31 public:
32 SNode() : next(nullptr) {}
33 };
34
35 template <typename T>
36 class SLinkedList {
37 public:
38 SLinkedList();
39 ~SLinkedList();
40 bool empty() const;
41 const T& front() const;
```

```
42 void addFront(const T& e);
43 void addBack(const T& e);
44 void removeFront();
45 friend void concat<T>(SLinkedList<T>& l1, SLinkedList<T>& l2,      ↗
    SLinkedList<T>&
46 lout);
47 private:
48 SNode<T>* head;
49 };
50
51 template <typename T>
52 void concat(SLinkedList<T>& l1, SLinkedList<T>& l2, SLinkedList<T>&      ↗
    lout); //Homework
53
54 template <typename T>
55 SLinkedList<T>::SLinkedList() : head(nullptr) { } // constructor
56
57 template <typename T>
58 bool SLinkedList<T>::empty() const {
59 return head == nullptr; // is list empty?
60 }
61
62 template <typename T>
63 const T& SLinkedList<T>::front() const {
64 return head->elem; // return front element
65 }
66
67 template <typename T>
68 SLinkedList<T>::~~SLinkedList() {
69 while (!empty()) removeFront(); // destructor
70 }
71
72 template <typename T>
73 void SLinkedList<T>::addFront(const T& e) { // Add node to front
74 SNode<T>* v = new SNode<T>;
75 v->elem = e;
76 v->next = head;
77 head = v;
78 }
79
80 template <typename T>
81 void SLinkedList<T>::addBack(const T& e) { // Add node to the back
82 SNode<T>* v = new SNode<T>;
83 v->elem = e;
84 SNode<T>* n = head;
85 if (head == nullptr){ // Empty list
86 head = v;
87 return;
88 }
```

```
89 while (n->next != nullptr) {
90     n = n->next;
91 }
92 n->next = v;
93 }
94
95 template <typename T>
96 void SLinkedList<T>::removeFront() { // Remove node from front
97     SNode<T>* old = head;
98     head = old->next;
99     delete old;
100 }
101
102 template <typename T>
103 void concat(SLinkedList<T>& l1, SLinkedList<T>& l2, SLinkedList<T>& lout) ↗
104 {
105     // Your code here
106     // Iterate through l1 and add its elements to lout
107     SNode<T>* current = l1.head;
108     while (current != nullptr) {
109         lout.addBack(current->elem);
110         current = current->next;
111     }
112     // Iterate through l2 and add its elements to lout
113     current = l2.head;
114     while (current != nullptr) {
115         lout.addBack(current->elem);
116         current = current->next;
117     }
118 }
119
120 // Test
121 int main() {
122     SLinkedList<std::string> p1;
123     SLinkedList<std::string> p2;
124     SLinkedList<std::string> p3;
125     // Add elements
126     p1.addBack("C");
127     p1.addBack("C++");
128     p1.addBack("Java");
129     p1.addBack("Python");
130     p1.addBack("Javascript");
131
132     p2.addBack("Go");
133     p2.addBack("Rust");
134     p2.addBack("Julia");
135
136     // Concatenate the progLangsNew list to the end of progrLangs list
```

```
137 concat(p1,p2,p3);
138
139 // Print the concatenated list by repeatedly removing from list
140 while (!p3.empty()) { // Should print C C++ Java Python Javascript Go Rust ➤
    Julia
141     std::cout << p3.front() << " ";
142     p3.removeFront();
143 }
144 std::cout << std::endl;
145 }
```