```
#include <iostream>

#include <unordered_set>

#include <string>

using namespace std;



struct Node {

   string name;

   Node* next;



   Node(string n) : name(n), next(nullptr) {}

};



class LinkedList {

   Node* head;



public:

   LinkedList() : head(nullptr) {}



   void add(string name) {

      Node* new_node = new Node(name);

      new_node->next = head;

      head = new_node;

   }



   unordered_set<string> to_set() {

      unordered_set<string> result;

      Node* temp = head;

      while (temp) {

         result.insert(temp->name);

         temp = temp->next;

      }

      return result;
```

```
    }

    void display() {
        Node* temp = head;
        while (temp) {
            cout << temp->name << " ";
            temp = temp->next;
        }
        cout << endl;
    }

    ~LinkedList() {
        while (head) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
    }
};

int main() {
    LinkedList vanilla;
    LinkedList butterscotch;

    // Adding students to Vanilla and Butterscotch lists
    vanilla.add("Alice");
    vanilla.add("Bob");
    vanilla.add("Charlie");
    butterscotch.add("David");
    butterscotch.add("Bob");
    butterscotch.add("Eve");
```

```
cout << "Students who like Vanilla: ";
vanilla.display();
cout << "Students who like Butterscotch: ";
butterscotch.display();

unordered_set<string> setA = vanilla.to_set();
unordered_set<string> setB = butterscotch.to_set();

// Set of students who like both vanilla and butterscotch
unordered_set<string> intersection;
for (const auto& student : setA) {
   if (setB.find(student) != setB.end()) {
      intersection.insert(student);
   }
}

cout << "\nStudents who like both Vanilla and Butterscotch: ";
for (const auto& student : intersection) {
   cout << student << " ";
}
cout << endl;

// Set of students who like either vanilla or butterscotch or not both
unordered_set<string> union_set = setA;
for (const auto& student : setB) {
   union_set.insert(student);
}

unordered_set<string> symmetric_difference;
for (const auto& student : union_set) {
   if (setA.find(student) == setA.end() || setB.find(student) == setB.end()) {
      symmetric_difference.insert(student);
```

```
        }
    }

    cout << "Students who like either Vanilla or Butterscotch or not both: ";
    for (const auto& student : symmetric_difference) {
        cout << student << " ";
    }
    cout << endl;

    // Number of students who like neither vanilla nor butterscotch
    int total_students = 10; // Assume there are 10 students in the class
    int neither_count = total_students - union_set.size();

    cout << "Number of students who like neither Vanilla nor Butterscotch: " << neither_count << endl;

    return 0;
}
```