The fixed code:

```
#include <cstdio>

int number_instantiated = 0;

class Node {
public:
   Node(const int &value, Node *next = nullptr)
      : value_(value), next_(next) {
      printf("%s%d%s\n", "Creating Node, ",
            ++number_instantiated,
            " are in existence right now");
   }

   ~Node() {
      printf("%s%d%s\n", "Destroying Node, ",
            --number_instantiated,
            " are in existence right now");
      next_ = nullptr;
   }

   Node *next() const { return next_; }
   void next(Node *new_next) { next_ = new_next; }
   const int &value() const { return value_; }
   void value(const int &value) { value_ = value; }

private:
   int value_;
```

```
  Node *next_;
};

class LinkedList {
public:
  LinkedList() : head_(nullptr) {}
  ~LinkedList() { delete_nodes(); }

  int insert(const int &new_item) {
    head_ = new Node(new_item, head_);
    return 0;
  }

  int remove(const int &item_to_remove) {
    Node *marker = head_;
    Node *temp = nullptr;
    bool item_removed = false;
    while (marker != nullptr) {
      if (marker->value() == item_to_remove) {
        if (temp == nullptr) {
          head_ = marker->next();
        } else {
          temp->next(marker->next());
        }
        Node *node_to_delete = marker;
        marker = marker->next();
        delete node_to_delete;
        item_removed = true;
        break;
```

```
            }
            temp = marker;
            marker = marker->next();
        }
        return item_removed ? 0 : -1;
    }

    void print() {
        Node *marker = head_;
        while (marker != nullptr) {
            printf("%d\n", marker->value());
            marker = marker->next();
        }
    }

private:
    void delete_nodes() {
        Node *marker = head_;
        while (marker != nullptr) {
            Node *temp = marker->next();
            delete marker;
            marker = temp;
        }
    }

    Node *head_;
};

int main(int argc, char **argv) {
    LinkedList list;

    list.insert(1);

    list.insert(2);

    list.insert(3);

    list.insert(4);

    printf("%s\n", "The fully created list is:");

    list.print();

    printf("\n%s\n", "Now removing elements:");

    list.remove(2);

    list.print();

    return 0;

}
```

Observation: In the remove function, the line delete temp; should be delete marker; because we want to delete the node that has the value equal to item_to_remove. Also, we should break the loop after removing a node.