```cpp
#include "IntList.h"
#include <iostream>

IntList::IntList() {
    this->count = 0;
    this->first = 0;
};

IntList::IntList(const IntList &other) {
    this->count = 0;
    this->first = 0;

    IntListElem *el;
    el = other.first;
    while (el) {
        this->insert(el->value, -1);
        el = el->next;
    }
};

IntList &IntList::operator=(const IntList &other) {
    if (this != &other) {
        while (this->getCount()) {
            this->remove(0);
        }

        IntListElem *el;
        el = other.first;
        while (el) {
            this->insert(el->value, -1);
            el = el->next;
        }
    }
    return *this;
};

IntList::~IntList() {
    if (this->getCount() > 0 && this->first != 0) {
        IntListElem *cur = this->first;
        IntListElem *next = cur->next;
        while (next) {
            cur->value = 0;
            delete cur;
            cur = next;
            next = cur->next;
        }
    }
};

int IntList::getCount() {
    return this->count;
};

bool IntList::isEmpty() {
    return this->getCount() > 0;
};


void IntList::print() {
    if (this->getCount()) {
        IntListElem *el = this->first;
        std::cout << "[" << el->value;
        el = el->next;
        while (el) {
            std::cout << "," << el->value;
            el = el->next;
        }
        std::cout << "]" << std::endl;
    } else {
        std::cout << "empty list" << std::endl;
    }
};

void IntList::insert(int element, int position) {
    if (position > 0 && position < this->getCount()) {
        IntListElem *elmToMod = this->first;
        for (int i = 0; i < position; i++) {
            elmToMod = elmToMod->next;
        }
        elmToMod->value = element;
    } else if (position == this->getCount()) {
        IntListElem *elmToInsert = new IntListElem;
        elmToInsert->value = element;
        elmToInsert->next = 0;
        IntListElem *elmAfter = this->first;
        if (position == 0) {
            this->first = elmToInsert;
        } else {
            while (elmAfter->next) {
                elmAfter = elmAfter->next;
            }
            elmAfter->next = elmToInsert;
```

```
 93         }
 94         this->count++;
 95     } else if (position == -1) {
 96         this->insert(element, this->getCount());
 97     }
 98 };
 99
100 void IntList::remove(int position) {
101     IntListElem *elToDel, *elPrev;
102     elToDel = findElement(position);
103     if (position == 0) {
104         if (elToDel) {
105             this->first = elToDel->next;
106             this->count--;
107         }
108     } else if (position > 0 && position < this->getCount()) {
109
110         findElement(position - 1)->next = elToDel->next;
111         this->count--;
112     }
113     delete elToDel;
114
115 };
116
117 int IntList::getElement(int position) {
118     IntListElem *el = findElement(position);
119     return el ? el->value : 0;
120 };
121
122 IntList::IntListElem *IntList::findElement(int position) {
123     IntListElem *el = this->first;
124     for (int i = 0; i < position && el; i++) {
125         el = el->next;
126     }
127     return el;
128 };
129
```