#include <iostream>

template <class T>

class list

{

public:

struct node

{

T value;

node \* next;

node(T v, node \* n):value(v), next(n){};

};

list():count(0), head(NULL), tail(NULL){};

~list()

{

node \*t;

for(t = head; t != NULL; t = t->next)

{

delete t;

}

}

void push(T value)

{

node \*t = new node(value, NULL);

if (head == NULL)

{

head = t;

tail = t;

}

else

{

tail->next = t;

tail = t;

}

count++;

}

node\* pop()

{

if(head == NULL)

{

return NULL;

}

node \*t = head;

head = head->next;

count--;

return t;

}

node\* search(T value)

{

if(head == NULL)

{

return NULL;

}

node \*t;

for(t = head; t != NULL; t = t->next)

{

if (t->value == value)

{

return t;

}

}

return NULL;

}

int size()

{

return count;

}

bool isEmpty()

{

return (count == 0);

}

private:

int count;

node \* head;

node \* tail;

};

int main(int argc, char\*\* argv) {

list<int> l;

l.push(25);

l.push(13);

l.push(-25);

l.push(-13);

list<int>::node \*t, \*t1;

t = l.search(13);

t1 = l.search(12);

if(t != NULL)

{

std::cout << "Value: " << t->value << std::endl;

}

else

{

std::cout << "Value is not search!" << std::endl;

}

if(t1 != NULL)

{

std::cout << "Value: " << t->value << std::endl;

}

else

{

std::cout << "Value is not search!" << std::endl;

}

while(!l.isEmpty())

{

t = l.pop();

std::cout << "Value: " << t->value << std::endl;

}

return 0;

}