```
#include <iostream>
using namespace std;

class SentinelLinkedList
{
	class Node
	{
	public:
		int value;
		Node *next;
		Node *prev;
		Node(int value)
		{
			this->value=value;
			next=nullptr;
			prev=nullptr;

		}
	};

	Node *head,*tail;
public:
	SentinelLinkedList()
	{
		Node *head=new Node(0);
		Node *tail=new Node(0);

		this->head=head;
		this->tail=tail;
		this->head->next=this->tail;
		this->tail->prev=this->head;
	}



void addToBack(int value)
{
	Node *NewNode=new Node(value);
	NewNode->next=tail;
	NewNode->prev=tail->prev;
	tail->prev->next=NewNode;
	tail->prev=NewNode;

}

void printForward()
{
	for(Node *current=head->next;current!=tail;current=current->next)
	{
		cout << current->value<<endl;
	}
}
```

```
void printBackward()
{
	for(Node *current=tail->prev;current!=head;current=current->prev)
	{
		cout << current->value<<endl;
	}


}
void addToFront(int value)
{
	Node *NewNode=new Node(value);

	NewNode->next=head->next;
	NewNode->prev=head;
	head->next->prev=NewNode ;
	head->next=NewNode;

}


void insertBefore(int search,int value)
{
	for(Node *current=head->next;current!=tail;current=current->next)
	{
		if(search==current->value)
		{
			Node *newNode=new Node(value);
			newNode->next=current;
			newNode->prev=current->prev;
			newNode->prev->next=newNode;
			newNode->next->prev=newNode;
			break;
		}
	}
}

bool removeNode(int value)
{
	for (Node *curr=head->next; curr!=tail; curr = curr->next)
	{
		if(value == curr->value)
		{
			curr->next->prev = curr->prev;
			curr->prev->next = curr->next;
			delete curr;
			return true;
		}
	}

	return false;
}

void insertAfter(int search,int value)
```

```
{
	for(Node *current=head->next;current!=tail;current=current->next)
	{
		if(search==current->value)
		{
			Node *newNode=new Node(value);

			newNode->next = current->next;
			newNode->prev = current;
			newNode->prev->next = newNode;
			newNode->next->prev = newNode;
			break;
		}
	}
}

void removeAll()
	{
		while  (head->next!=tail)
		{
			head->next = head->next->next;
			delete head-> next->prev;
		}

		tail->prev = head;
}


};


int main()
{

	SentinelLinkedList s1;
	int num,value;

	while (cout << "enter value (enter 0 to stop)" << endl, cin >> num,
num)
	{
		s1.addToBack(num);
	}

	cout << "Printing forwrd" << endl;
	s1.printForward();

	cout << "Print Backward" << endl;
	s1.printBackward();

	while (cout << "enter the value you want to enter in front (enter 0
to stop)" << endl, cin >> num, num)
	{
		s1.addToFront(num);
	}
```

```
	s1.printForward();

	while (cout << "enter the value before which you want to add new
value (enter 0 to stop)" << endl, cin >> num, num)
	{
		cout << "enter value you want to add"<<endl;
		cin >> value;
		s1.insertBefore(num,value);
	}
	s1.printForward();

	while (cout << "enter the value after which you want to add new
value (enter 0 to stop)" << endl, cin >> num, num)
	{
		cout << "enter value you want to add"<<endl;
		cin >> value;
		s1.insertAfter(num,value);
	}
	s1.printForward();

	while (cout << "enter the value you want to enter in remove (enter
0 to stop)" << endl, cin >> num, num)
	{
		s1.removeNode(num);
	}
	s1.printForward();

	cout << "remove all node" << endl;
	s1.removeAll();
	s1.printForward();

}
```