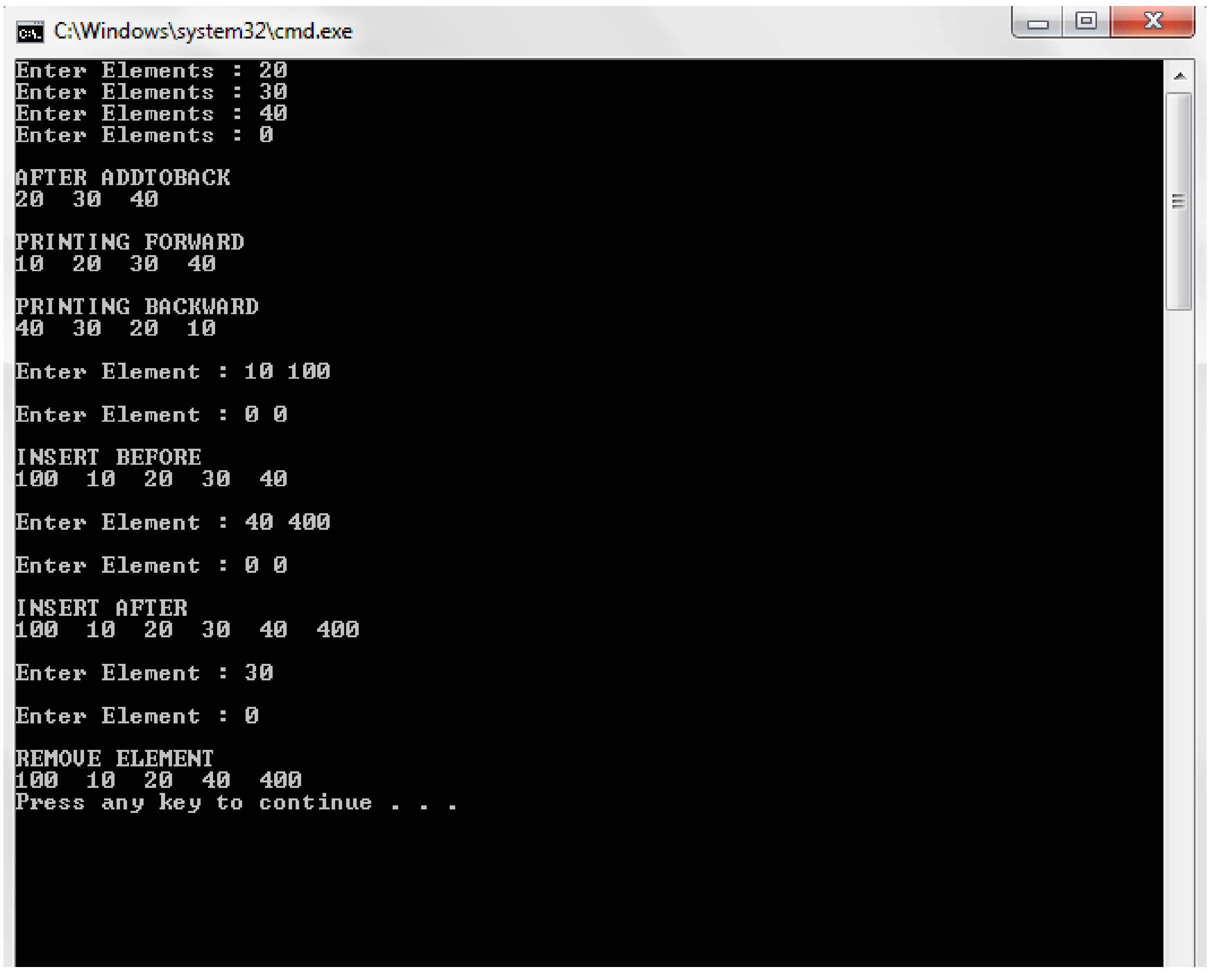
CIRCULAR SENTINEL DOUBLE LINKED LIST

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

class Circular_Sentinel
{
private:
	struct Node
	{
		Node *prev;
		int value;
		Node *next;

		Node (int value)
		{
			this -> prev = nullptr;
			this -> value = value;
			this -> next = nullptr;
		}

	};

	Node *head;
	Node *tail;

public:

	Circular_Sentinel ()
	{
		head = new Node(0);
		tail = new Node(0);
		head -> next = tail;
		head -> prev = tail;
		tail -> prev = head;
		tail -> next = head;
	}

	void insert (Node *current, int value)
	{
		Node *newnode = new Node(value);
		newnode->next=current;
		newnode->prev=current->prev;
		newnode->next->prev=newnode->prev->next=newnode;
	}

	void addToBack(int value)
	{
		insert(tail,value);
	}

	void addToFront(int value)
	{
		insert(head->next,value);
	}
```

```
        bool insertAfter(int search, int value)
        {
                for(Node *p =head->next; p!=tail; p=p->next)
                {
                        if(search==p->value)
                        {
                                insert (p->next,value);
                                return true;
                        }
                }
                return false;
        }

        bool insertBefore(int search, int value)
        {
                for(Node *p=head->next; p!=tail; p=p->next)
                {
                        if(search==p->value)
                        {
                                insert(p,value);
                                return true;
                        }
                }
                return false;
        }

        void printforward ()
        {
                for(Node *p=head->next; p!=tail; p=p->next)
                {
                        cout<<p->value<<"  ";
                }
                cout<<endl;
        }

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

        bool remove(int search)
        {
                for(Node *p=head->next; p!=tail; p=p->next)
                {
                        if(search==p->value)
                        {
                                p->prev->next=p->next;
                                p->next->prev=p->prev;
                                delete p;
                                return true;
                        }
                }
                return false;
        }

};
```

```cpp
int main()
{
	Circular_Sentinel cs;
	int search;
	int num;

	while(cout<< "Enter Elements : " , cin>> num, num)
	{
		cs.addToBack(num);
	}


	cout<<" \nAFTER ADDTOBACK "<<endl;
	cs.printforward();

	cs.addToFront(10);

	cout<<"\nPRINTING FORWARD "<<endl;
	cs.printforward();

	cout<<"\nPRINTING BACKWARD"<<endl;
	cs.printBackward();

	while(cout<<"\nEnter Element : ", cin>>search>>num, num)
	{
		cs.insertBefore(search,num);
	}

	cout<<"\nINSERT BEFORE"<<endl;
	cs.printforward();

	while(cout<<"\nEnter Element : ", cin>>search>>num, num)
	{
		cs.insertAfter(search,num);
	}

	cout<<"\nINSERT AFTER"<<endl;
	cs.printforward();

	while(cout<<"\nEnter Element : ", cin>>search, search)
	{
		cs.remove(search);
	}

	cout<<"\nREMOVE ELEMENT"<<endl;
	cs.printforward();

}
```



```
Enter Elements : 20
Enter Elements : 30
Enter Elements : 40
Enter Elements : 0

AFTER ADDTOBACK
20  30  40

PRINTING FORWARD
10  20  30  40

PRINTING BACKWARD
40  30  20  10

Enter Element : 10 100

Enter Element : 0 0

INSERT BEFORE
100  10  20  30  40

Enter Element : 40 400

Enter Element : 0 0

INSERT AFTER
100  10  20  30  40  400

Enter Element : 30

Enter Element : 0

REMOVE ELEMENT
100  10  20  40  400
Press any key to continue . . .
```