```
/****************************************************************************************************
**
**	Name: Clyde Pabro
**	Class:		CISP430 - Fall 2012 Thu
**	Assignment:	Homework 5
**
****************************************************************************************************/
```

Part 1:

A. Wrote List ADT using Link Nodes. Source code following time complexities (Part 2).
B. Wrote List ADT using Circular Arrays. Source code following Part 2.

Part 2:

Time Complexities for List ADT using Link Nodes

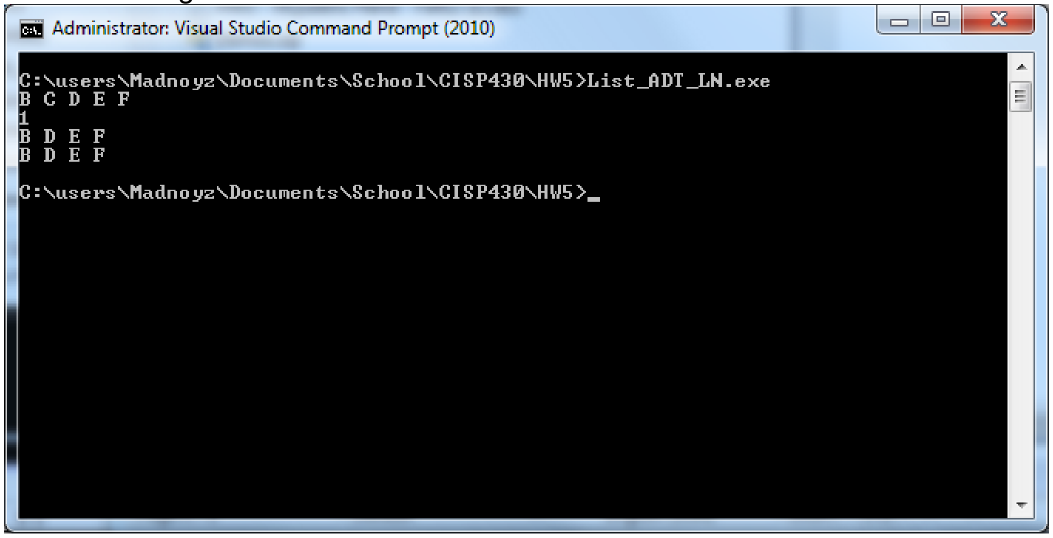
Function Time Complexity
Time complexity of isEmpty()............O(1) Reason: It only includes an if..then statement and does not loop.
Time complexity of traverse()........... O(n) Reason: It runs in a loop while it still has a head.
Time complexity of append().............O(1) Reason: Does not loop. Only includes an if..then statement.
Time complexity of removeData().... O(1) Reason: Does not loop. Only includes if...then statements.
Time complexity of find()..................O(n) Reason: Includes a while loop searching through the nodes.

Time Complexities for List ADT using Circular Arrays

Function Time Complexity
Time complexity of isEmpty()............O(1) Reason: It only includes an if..then statement and does not loop.
Time complexity of traverse()........... O(n) Reason: It runs in a loop while it still has a head.
Time complexity of append().............O(1) Reason: Does not loop. Only includes an if..then statement.

Time complexity of removeData().... O(1) Reason: Does not loop. Only includes if...then statements.
Time complexity of find()..................O(n) Reason: Includes a while loop searching through the nodes.

OUTPUT Program 1:



```
C:\users\Madnoyz\Documents\School\CISP430\HW5>List_ADT_LN.exe
B C D E F
1
B D E F
B D E F

C:\users\Madnoyz\Documents\School\CISP430\HW5>_
```

SOURCE CODE Program 1.

```
/**********************************************
List ADT using Link Nodes Source Code:
**********************************************/
#include <iostream>
#include <cstdlib>
using namespace std;

typedef char data;

struct node {
        data d;
        node *next;
};

node *head = 0;
node *tail = 0;

int isEmpty( void );
void traverse( void );
void append( data );
data removeData( void );
int find( char );
int main()
{
	append('A');
	append('B');
	append('C');
	append('D');
	append('E');
	append('F');
	removeData();
	traverse();
	cout << find('C') << " ";
	cout << endl;
	traverse();
	while( !isEmpty() )
	{
		cout << removeData() << " ";
	}
	cout << endl;
	return 0;
}

int isEmpty( void )
{
	if( head )
		return 0;
	else
		return 1;
}

void traverse( void )
{
	node *p = head;

	while( p )
	{
		cout << (char)p->d << " ";
		p = p->next;
	}
	cout << endl;
}

void append( data d )
{
	node * p = (node*)malloc(sizeof(node));
	p->next = 0;
	p->d = d;

	if( !head )
	{
		head = tail = p;
```

```
        } else {
                tail->next = p;
                tail = p;
        }
}

data removeData( void )
{
        node *p;
        data temp;

        if( !head )
                return -1;

        if( head == tail )
        {
                temp = head->d;
                free (head);
                head = tail = 0;
                return temp;
        }

        p = head;
        head = head->next;
        temp = p->d;
        free (p);
        return temp;
}

int find( data d )
{
        node *c;
        node *pc;

        if( !head )
                return 0;

        if( head == tail )
        {
                if( head->d == d )
                {
                        free (head);
                        head = tail = 0;
                        return 1;
                }else{
                        return 0;
                }
        }

        pc = head;
        c = head->next;
```

```
	if( pc->d == d )
	{
		head = head->next;
		free (pc);
		return 1;
	}

	while( c )
	{
		if( c->d == d )
		{
			pc->next = c->next;

			if( c == tail )
				tail = pc;

			free (c);
			return 1;
		}

		pc = c;
		c = c->next;
	}
	return 0;
}
```

OUTPUT Program 2:

```
/**************************************************
List ADT using Circular Array Source Code:
**************************************************/
#include <iostream>
#include <cstdlib>
#define SIZE 4
using namespace std;

typedef int data;

data myList[SIZE];
int head, tail, used, temp;

int isEmpty( void );
void append( data d );
void traverse( void );
data removeData( void );
int find( data d );

int main()
{
        head = tail = used = 0;
        append('A');
        append('B');
        append('C');
        append('D');
        append('E');
        append('F');
        removeData();
        traverse();
        cout << find('C') << endl;
        traverse();
        while( !isEmpty() )
        {
                cout << removeData() << " ";
        }
        cout << endl;
        return 0;
}

int isEmpty( void )
{
        if( used )
                return 1;
        else
                return 0;
}

void append( data d )
{
        if( !used )
        {
                myList[0] = d;
                used++;
                return;
        }

        tail = ( tail + 1 ) % SIZE;
        myList[ tail ] = d;
        used++;
}

void traverse( void )
{
        data p;

        if( isEmpty() == 0 )
        {
                cout << "Empty" << endl;
                return;
        }
        if( used == 1 )
        {
                cout << myList[ head ] << " " << endl;
                return;
        }

        p = head;
        do
        {
                cout << myList[p] << " ";
                p = ( p + 1 ) % SIZE;
        }while( p != ( tail + 1 ) % SIZE );
        cout << endl;
}

data removeData( void )
{
        if( isEmpty() )
                return -1;

        if( used == 1 )
        {
                used = 0;
                return myList[ head ];
        }

        temp = myList[ head ];
        head = ( head + 1 ) % SIZE;
        used--;
        return temp;
}

int find( data d )
{
        int p;

        if( isEmpty() == 0 )
                return 0;

        if( used == 1 )
        {
                if( myList[ head ] == d )
                {
                        used = 0;
                        return 1;
                }else{
                        return 0;
                }
        }

        p = head;
        do
        {
                if( myList[ p ] == d )
                {
                        while( p != tail )
                        {
                                myList[ p ] = myList[ ( p + 1 ) % SIZE ];
                                p = ( p + 1 ) % SIZE;
                        }

                        tail--;
                        if( tail < 0 )
                                tail = SIZE - 1;

                        used--;
                        return 1;
                }
        }while( p != (tail + 1) % SIZE );
}
```