CS 696 Mobile Application Development
Fall Semester, 2010
Doc 11 Assignment 1 Comments Part 1
Oct 5, 2010

No rar files

Really I mean it

No rar files

Names

```
@protocol Linked_listp
@interface coll
@interface node
@interface myListNode
```

```
static int mycounter = 0;

-(id) addtolinklist : (id) arrValue

nextnodeval

-(Node *) addf: (id) item1;
-(Node *) addl: (id) item1;
-(void) remf;
-(void) reml;
```

```
- (NSNumber*)next
{     int n3= [self intValue];
      count=count+1;
      NSNumber *n2=n3+1;
      return n2;
            }
```

Type Checking

```
- (NSNumber*)next
{     int n3= [self intValue];
      count=count+1;
      NSNumber *n2=n3+1;
      return n2;
            }
```

Compiler warns you about type issues

Compiler does not force types to be correct

Types are check at runtime

If messages are correct then no error

Spot the Warning?

⚠ Initialization makes integer from pointer without a cast
⚠ Passing argument 1 of 'objectAtIndex:' makes pointer from integer without a cast
⚠ 'slinklist' may not respond to '-initWithArray:'
(Messages without a matching method signature will be assumed to return 'i… more
⚠ 'slinklist' may not respond to '+listWithArray:'
⚠ 'slinklist' may not respond to '-componentsJoinedByString:'
⚠ 'slinklist' may not respond to '-componentsJoinedByString:'
⚠ 'slinklist' may not respond to '-enumerateObjectsUsingBlock:'

⚠ Initialization makes pointer from integer without a cast
⚠ Return makes pointer from integer without a cast

⚠ Conflicting types for '-(void)setnext:(node *)nex'
⚠ Previous declaration of '-(BOOL)setnext:(node *)nex'

⚠ Comparison between pointer and integer
⚠ 'NSException' may not respond to '+raise:'
(Messages without a matching method signature will be assumed to return 'i… more
⚠ Comparison between pointer and integer
At top level:
⚠ Incomplete implementation of class 'slinklist'
⚠ Method definition for '-fileToDictionary:' not found

⚠ '@end' missing in implementation context
⚠ '@end' missing in implementation context
⚠ Passing argument 3 of 'block' makes pointer from integer without a cast

```
NSMutableDictionary * fileToDictionary(NSString *str) {
    NSString *lineOfFile;
    NSString* filePath = str;
    NSArray *arrayOfLines;
    BOOL valueFlag = TRUE;
    BOOL flag = FALSE;
    NSString *value,*key;
                if (valueFlag) {
                    value = tmp;
                    valueFlag =
FALSE;
                }
```

```
-(void)enumerateObjectsUsingBlock:(void(^)(id obj, NSUInteger index, BOOL *stop)) block
{
	Node *temp=head;
	int c=[count intValue];
	int v=0;
	id obj;
	NSUInteger index;
	BOOL *stop;
	while([temp data]!=nil)
	{
		obj=[temp data];
		index=v;
		stop=YES;
		if( v=c)
			stop=NO;
		block([temp data],index,stop);
		temp=[temp  ptr];
		v=v+1;
	}
}
```

```
@protocol Linkedlistprotocol
@property(readonly) NSNumber* count;
-(id)addFirst:(id)value2;
-(id)addLast: (id)value3;
-(void)removeFirst;
-(void)removeLast;
@end
```

```
-(void) removeLast{

        Node * temp=head;
        Node * temp2=[head  ptr];
        while([temp2  ptr]!=nil)
        {
                temp=[temp  ptr];
                temp2=[temp2  ptr];
        }
        [temp putlink:nil];
}
```

```
- (void) removeFirst {
        ListNode *tmp,*toRemove;
        toRemove = head;
        tmp = head;
        tmp = tmp->next;
        head = tmp;
        int i = [count intValue];
        i = i-1;
        self->count = [NSNumber numberWithInt:i];
        free (toRemove);
}
```

```
- (void) removeFirst {
        ListNode *toRemove = head;
        head = head->next;
        int nextCount = [count intValue] + 1;
        self->count = [NSNumber numberWithInt: nextCount];
      [toRemove->data release];
        free (toRemove);
}
```

Names

Use complete words - no abbreviations

Class

names start with Uppercase

method names, arguments, local variables, instance variables

Start with lowercase

afterFirstWordEachWordStartsUppercase

Method Naming Guidelines

Choose method names so that statements containing the method read like a sentence

[FileDescpriptor seekTo: work from: self position]

Use imperative verbs and phrases for methods which perform an action

[aDog sit]

[aFace lookSuprised]

Examples from Smalltalk With Style, page 10-11

Method Naming Guidelines

Use a phrase beginning with a verb (is, has) when a method returns a boolean

isString

aPerson isHungry

~~aPerson hungry~~

Use common nouns for methods which answer a specific object

anAuctionBlock nextItem

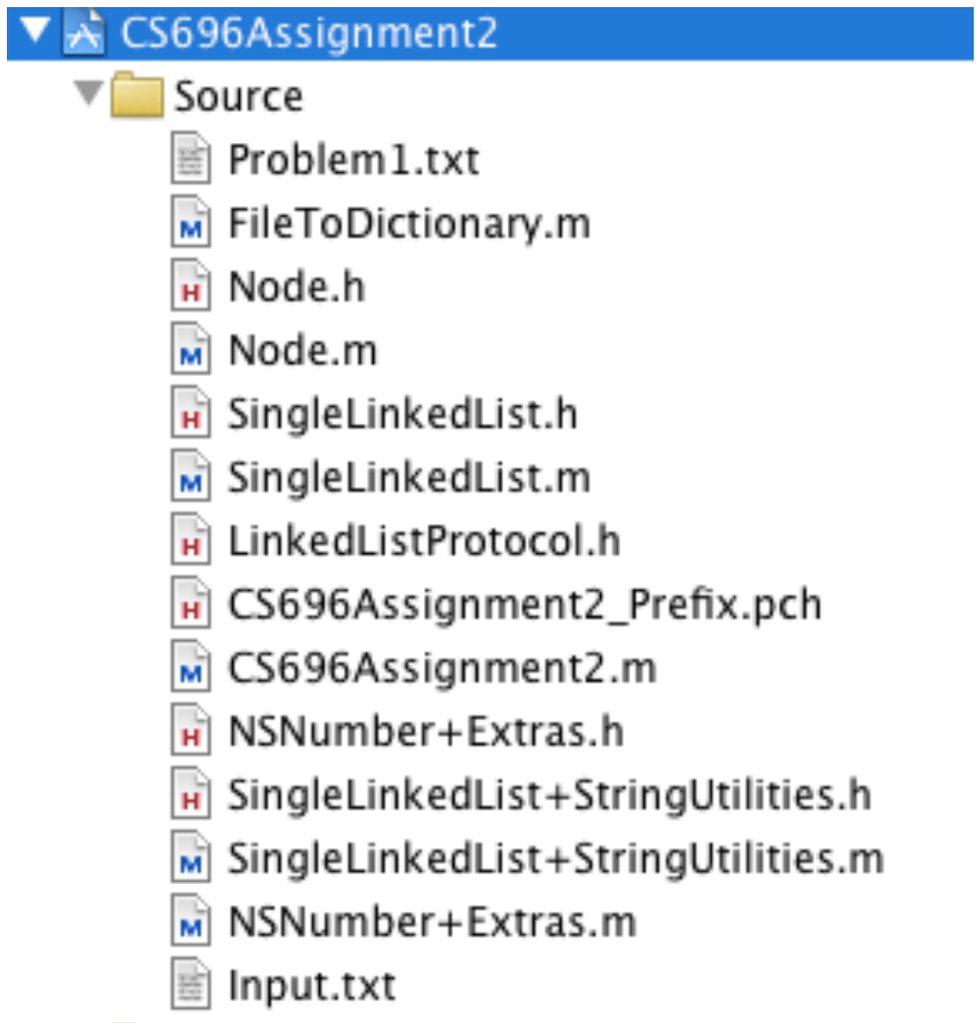
~~anAuctionBlock item~~ "which item"

From Smalltalk With Style, page 11-12

Formatting

```
- (NSString*)componentsJoinedByString: (NSString*) seperator
{
        NSMutableString *appendedString = [[NSMutableString alloc]init];

        Linkedlistnode *n;

        n=head;

        // appending separator to head node

        appendedString = [appendedString stringByAppendingFormat: @"%@",n->linklistvalue];
        n= n->next;

        // appending separator to nodes other than head and tail

        while(n->next!=NULL)
        {
                appendedString =[appendedString stringByAppendingFormat : @"%@",seperator];


                appendedString =[appendedString stringByAppendingFormat : @"%@",n->linklistvalue];
                n = n->next;
        }

        // appending separator to tail node

        appendedString =[appendedString stringByAppendingFormat : @"%@",seperator];
        appendedString =[appendedString stringByAppendingFormat: @"%@",n->linklistvalue];




        return appendedString;
}
```

```
while ([theScanner isAtEnd] == NO)
{
        if ([theScanner scanString: @"name: " intoString: NULL] && [theScanner scanUpToString: @"," intoString: &name] && [theScanner scanString: @"," intoString: NULL] && [theScanner scanString: @"phone: " intoString: NULL] && [theScanner scanUpToString: @"\n" intoString: &phone] )
        {
                [dict setValue:name forKey:phone];
        }
}
        }
}
```


CS696Assignment2
Source
Problem1.txt
FileToDictionary.m
Node.h
Node.m
SingleLinkedList.h
SingleLinkedList.m
LinkedListProtocol.h
CS696Assignment2_Prefix.pch
CS696Assignment2.m
NSNumber+Extras.h
SingleLinkedList+StringUtilities.h
SingleLinkedList+StringUtilities.m
NSNumber+Extras.m
Input.txt


Source
slinklist.h
slinklist.m
ques5.h
ques5.m
protocol.h
test_Prefix.pch
next.h
q4p6.h
q4p6.m
node.h
node.m
next.m
test.m
ques6.h
ques6.m

```
@interface NSNumber(newMethod)

- (NSNumber*)next;
      + (int)callCount;

@end
```

```
@implementation Node
        -(id)data{return item;}
        -(void)putdata:(id) x{item=x;};
        -(Node *)ptr{return link;};
        -(void)putlink:(Node *)x{link=x;};

@end
```

```
-(id)addFirst                : (id) input;
-(id)addLast                 : (id) input;
-(void)removeLast                        ;
-(void)removeFirst                       ;
```

```
	NSString *stringDescription=@"(";
	while ([temp getLink]!=nil) {

		stringDescription=[NSString stringWithFormat:@"%@ %@,",stringDescription,[temp
getValue]];
		temp=[temp getLink];
	}
	return [NSString stringWithFormat:@"%@ %@)",stringDescription,[temp getValue]];
}
-(id) addFirst:(id)value2
{
	//NSLog(@"after setData%i",(id)item1);

	if(head == nil)
	{
		//NSLog(@"No values in the linked list to add before front");
		[value2 retain];
		head= [[Node alloc] init];
		[head setValue:value2];
```

```
@implementation FiletoDictionary
void filetoDictionary(NSString *path)
{
    NSMutableDictionary *addtoDictionary = [[NSMutableDictionary alloc] init];

    NSError *error;

    NSString *stringFromFileAtPath = [NSString stringWithContentsOfFile: path encoding: NSUTF8StringEncoding error:&error];

    if (stringFromFileAtPath == nil)

    {

        NSLog(@"Error reading file at %@\n%@",path, [error localizedFailureReason]);

}

NSScanner *theScanner;

NSString *name;

NSString *phone;
```

```
int c=0;
- (NSNumber*)next: (int)j: (NSNumber*)pool
{
        int k=j;
        NSNumber *boo=pool;
        int* x;
        switch(k)
        {

                case 11:

                        NSLog(@"Before call count the value is %@",pool);

        boo=[NSNumber numberWithInt:[self intValue]+1];

                        x=(int*)boo;

        NSLog(@"next value is %@",x);

                        c++;
                        break;
        }

        return 0;

}
```

```
-(NSString *)componentsJoinedByString:(NSString *)seperator{
        [self getFront];
        outputString = @"";
        for (int i = 0; i < [self size]; i ++){
                NSLog(@"count:  %i",i);
                NSLog(@"value: %@", [current data]);
                outputString = [outputString stringByAppendingString : [current data]];
                if (i < [self size] - 1){
                outputString = [outputString stringByAppendingString : seperator];
                }
                [self getNext];
        }
        return outputString;
}
```

```
@interface Node: NSObject {
      id value;
      Node  *link;
}

-(Node *)addFrontNode:(id)value1;
-(Node *)addLastNode:(id)valu1;

@property(getter=getValue,readwrite,assign) id value;
@property(getter=getLink,readwrite,assign) Node *link;
@end

@interface Linkedlist : NSObject <Linkedlistprotocol> {

      Node *head;
}
```


Why in one file?

```
-(void)enumerateObjectsUsingBlock: (void (^)(id obj, NSUInteger idx, BOOL* stop))findAs {

        id data;
        NSUInteger index;
        BOOL stopNow = NO;

        [self getHead];

//      for (int x=0; [self getSize]-1 > x; x++) {
//              data = [self getCurrent];
//              index = x;
//              findAs(data, index, &stopNow);
//              [self getNext];
//      }

        int x=0;
        while ([self getSize]-1 > x && stopNow == NO) {
                data = [self getCurrent];
                index = x;
                findAs(data, index, &stopNow);
                [self getNext];
                x++;
        }
```

WTF

-(void)objectAtIndex: (NSInteger)Index: (int)q

```
-(void)addFirst: (int)y  {
      int i=y;
   switch(i) {
         case 2: if(h==NULL) {
               printf("\nList does not exist!!!!! \n");
               break;
      }
               printf("\nEnter the position: \n");
               scanf("%d",&pos);
               if(pos==0) {
                              /* If position is 0 (first element) then a new node is created
*/
                                    t=new_node;
                        t->link=h;
                        h=t;
                        count=[NSNumber numberWithInt:[count intValue]+1];
                     }
               printf("\Enter an integer element:\n");
               scanf("%d",&t->data);
               break;
      }
}
```

```
id x= [anArray lastObject];
//NSLog(@"%@",x);
NSUInteger cnt= [anArray indexOfObject:x]+1;
```

```
+(id)listWithArray:(NSArray*) anArray{


        Linkedlist *list1 = [[Linkedlist alloc] init];
        Node *link=nil;
        Node *temp2;
        id x= [anArray lastObject];
        NSUInteger cnt= [anArray indexOfObject:x]+1;
        if(cnt!=1)
        {
                for(int i=0;i<cnt;i++)
                {
                        temp2=[[Node alloc]init];
                        id val= [anArray objectAtIndex:cnt-i-1];
                        [temp2 setitem:val];
                        [temp2 setlink:link];
                        if(i==cnt-1)
                        {
                                list1->firstnode=temp2;
                        }
                           link=temp2;

                }
                return list1;
}

else
                {
                        temp2= [[Node alloc]init];
                        [temp2 setitem:[anArray objectAtIndex:0]];
                        list1->firstnode=temp2;
                        return list1;
                 }
}
```

```
-(id) addFirst:(id) node_data
{
	if(header_node == nil)
	{
		//header_node = [q3_node new];
		header_node = [[q3_node alloc]init];
		//header_node = node_data; - this id done by the node fucntion set_node_data
		[header_node set_node_data:node_data];
		count =[NSNumber numberWithInt:0];
	}
	else
	{
		// to add a new node we call add_firstnode from the node.m
		header_node=[header_node add_firstnode:node_data];
	}
	count =[NSNumber numberWithInt:[count intValue]+1]; // using the property variable
count
	return node_data;
}
```

Comments

```
// getters and setters
-(id) data {
	return data;
}

-(void) setData:(id)newData {
	if (data != newData) {
		[data release];
		data = [newData retain];
	}
}
```

Memory Management is hard

Hardest thing about the assignment

Very hard to figure out what you were doing

No one commented on how their model for memory management

It was the one that needed commenting

Memory

You own an object you it create using:

- method starting with new
- method starting with alloc
- method containing copy

You own an object if you sent it the message

- retain

When done with an object you own release it using

- release
- autorelease

Don't release it

When you

- have a reference to an object and
- need to keep the object

For example

- Instance variables
- Globals (static)

Collections retain objects when added

```
NSArray *sample;
NSDate *aDate = [NSDate date];
NSValue *aValue = [[NSNumber alloc] initWithInt:5];
NSString *aString = @"a string";

sample = [NSArray arrayWithObjects: aDate, aValue, aString, nil];
[aValue release];

NSDate * result =[sample objectAtIndex: 0];
NSLog(@"date: %@", result);

NSInteger size = [sample count];
```

Retain objects added to list

Make sure nodes have retain count > 0

remove
    release object in node
    release node

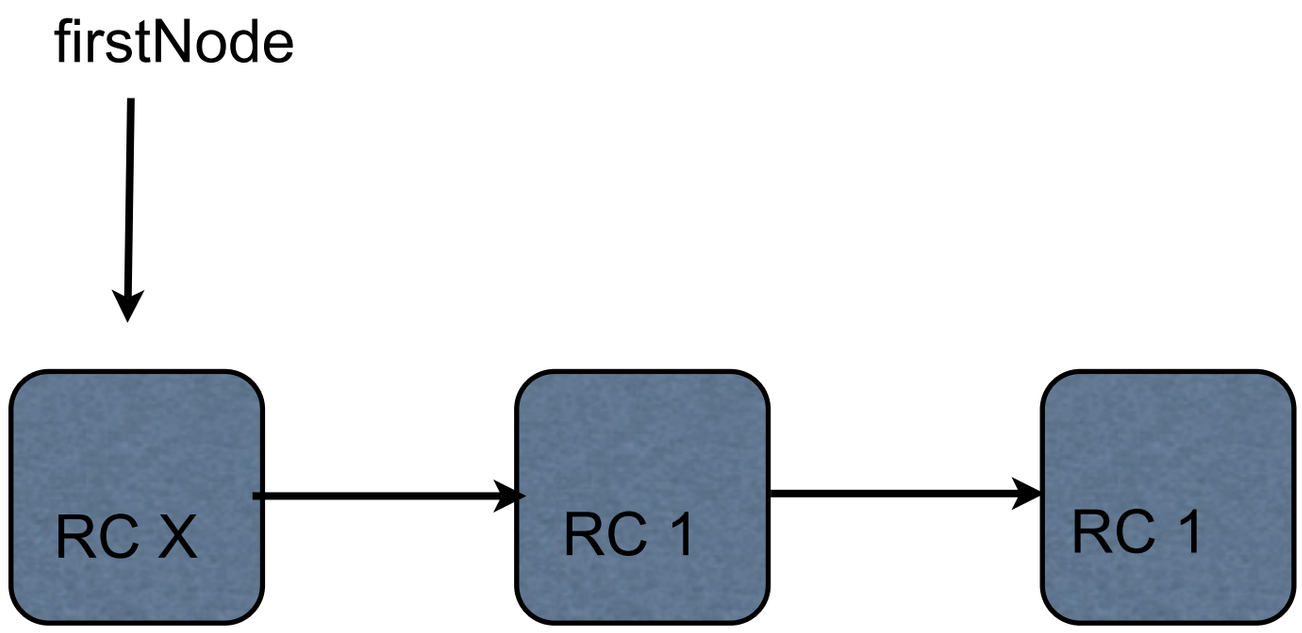
```
arrayOfLines = [[NSArray alloc] init];
arrayOfLines = [lineOfFile componentsSeparatedByCharactersInSet:s];
```

```
count=[[NSNumber alloc]initWithInt:[count intValue]+1];
[count release];                    //release memory from previous value
```

```
NSNumber *newNumber = [NSNumber numberWithInt:currentNum + 1] ;
[newNumber autorelease];
```

```
[retVal release];
return retVal.data;


[node release];
node = [node next];
```

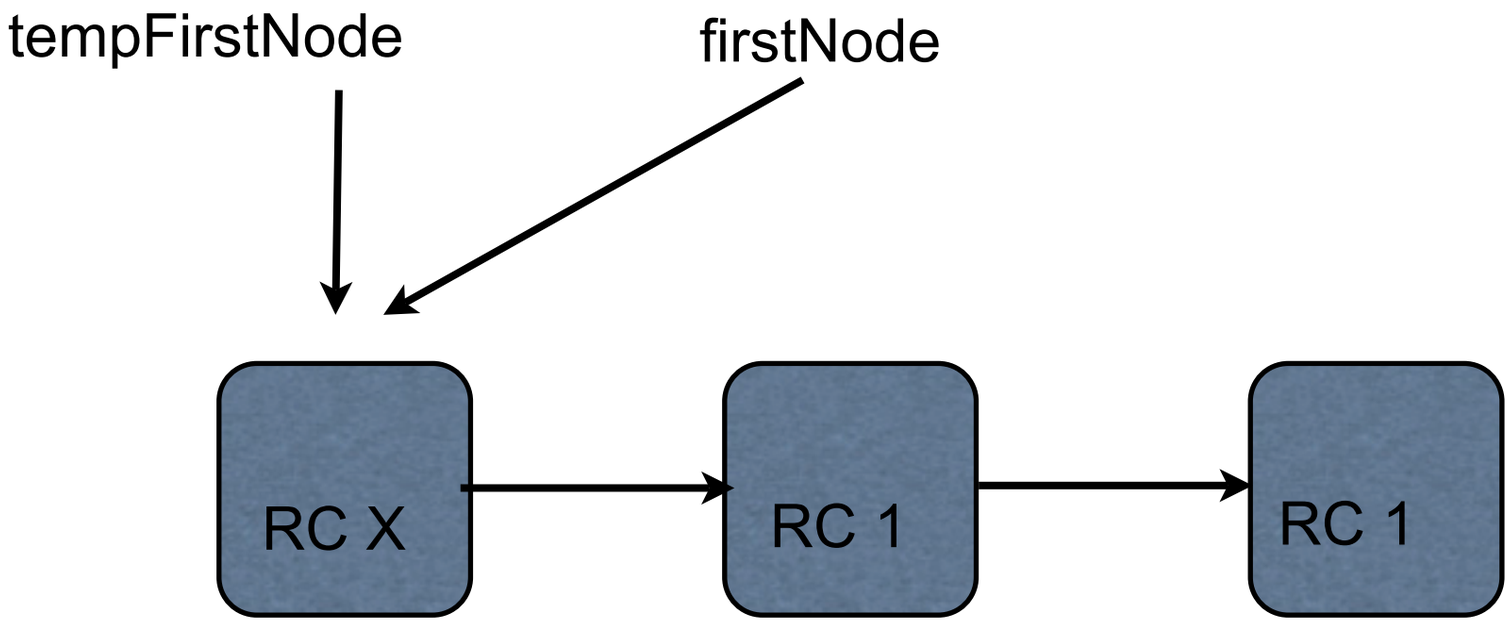
```
Node * tempFirstNode = firstNode;
[tempFirstNode retain];
[firstNode release];
firstNode = [tempFirstNode link];
[firstNode retain];
[tempFirstNode release];
count = [NSNumber numberWithInt:[[self count] intValue]- 1];
```



Slo Mo

No Effect

```
Node * tempFirstNode = firstNode;
[tempFirstNode retain];
[firstNode release];
firstNode = [tempFirstNode link];
[firstNode retain];
[tempFirstNode release];
count = [NSNumber numberWithInt:[[self count] intValue]- 1];
```

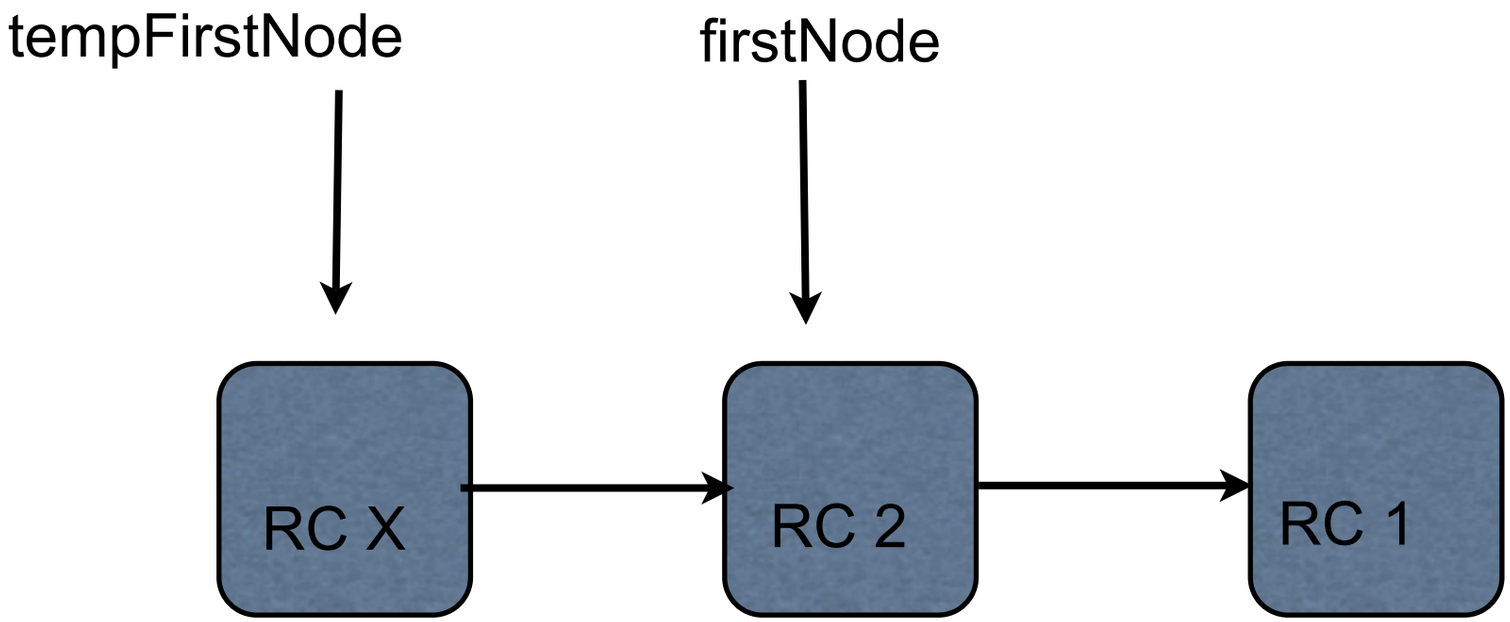


Slo Mo

```
Node * tempFirstNode = firstNode;
[tempFirstNode retain];
[firstNode release];
firstNode = [tempFirstNode link];
[firstNode retain];
[tempFirstNode release];
count = [NSNumber numberWithInt:[[self count] intValue]- 1];
```

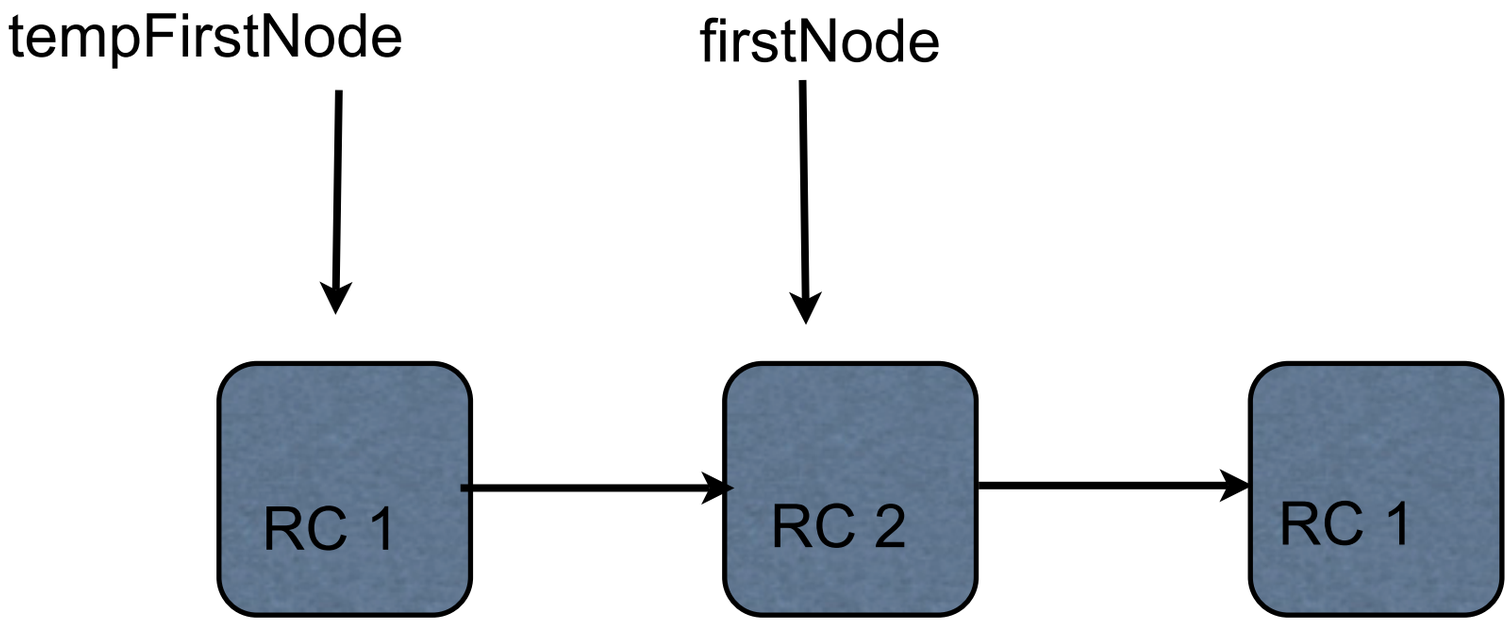
firstNode retain count is 2

So X = 2



Slo Mo

```
Node * tempFirstNode = firstNode;
[tempFirstNode retain];
[firstNode release];
firstNode = [tempFirstNode link];
[firstNode retain];
[tempFirstNode release];
count = [NSNumber numberWithInt:[[self count] intValue]- 1];
```



Trouble trouble

```
@interface Node : NSObject { }
@property (retain) id data;
@property (retain) Node *next;

@end
```

```
@implementation Node

@synthesize next;
@synthesize data;

- (void) dealloc{
      [data release];
      [next release];
      [super dealloc];
}

@end
```

LinkedList class

```
- (void) removeFirst{
      Node* temp = head;
      head = head.next;
      count = [NSNumber numberWithInt: ([coun
      [temp release];
}
```

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

Slo Mo

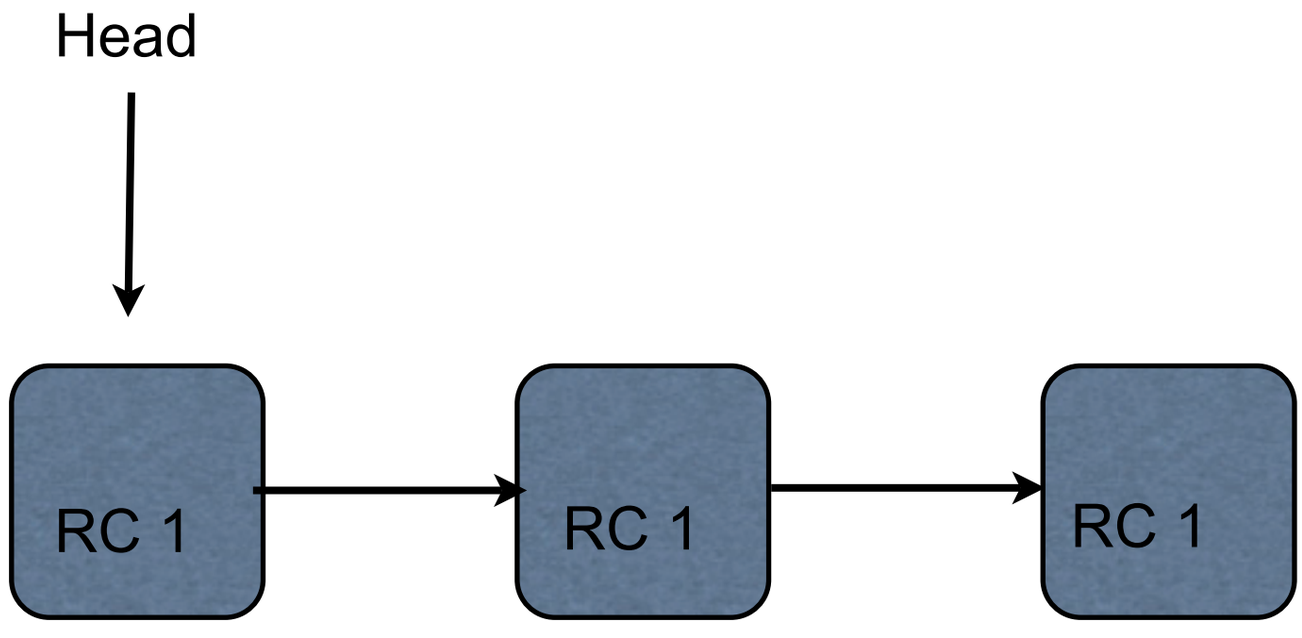
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        [temp release];
}
```



Slo Mo

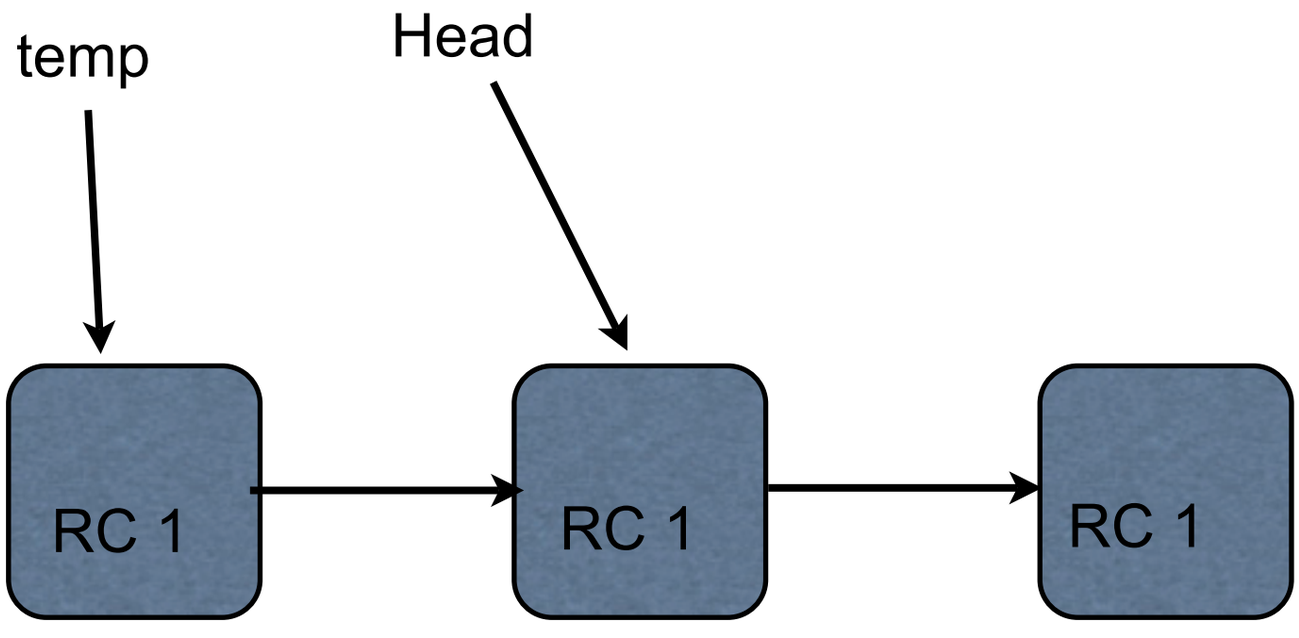
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        [temp release];
}
```



Slo Mo

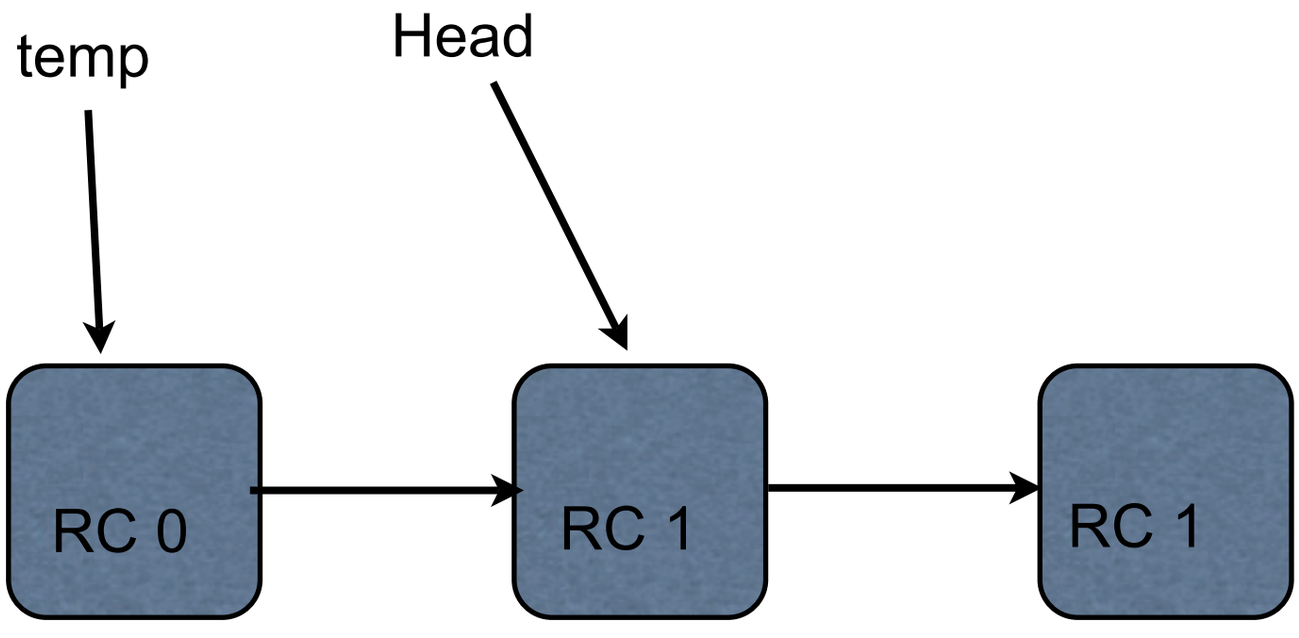
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        [temp release];
}
```



Slo Mo

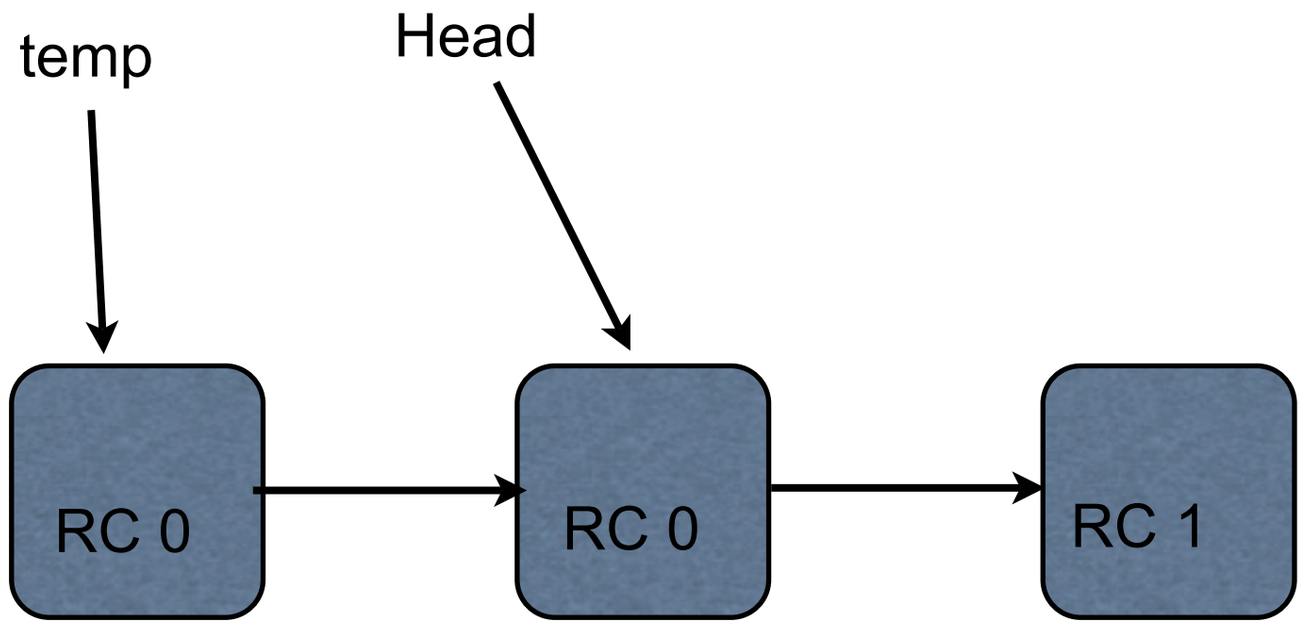
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        [temp release];
}
```



Slo Mo

Node

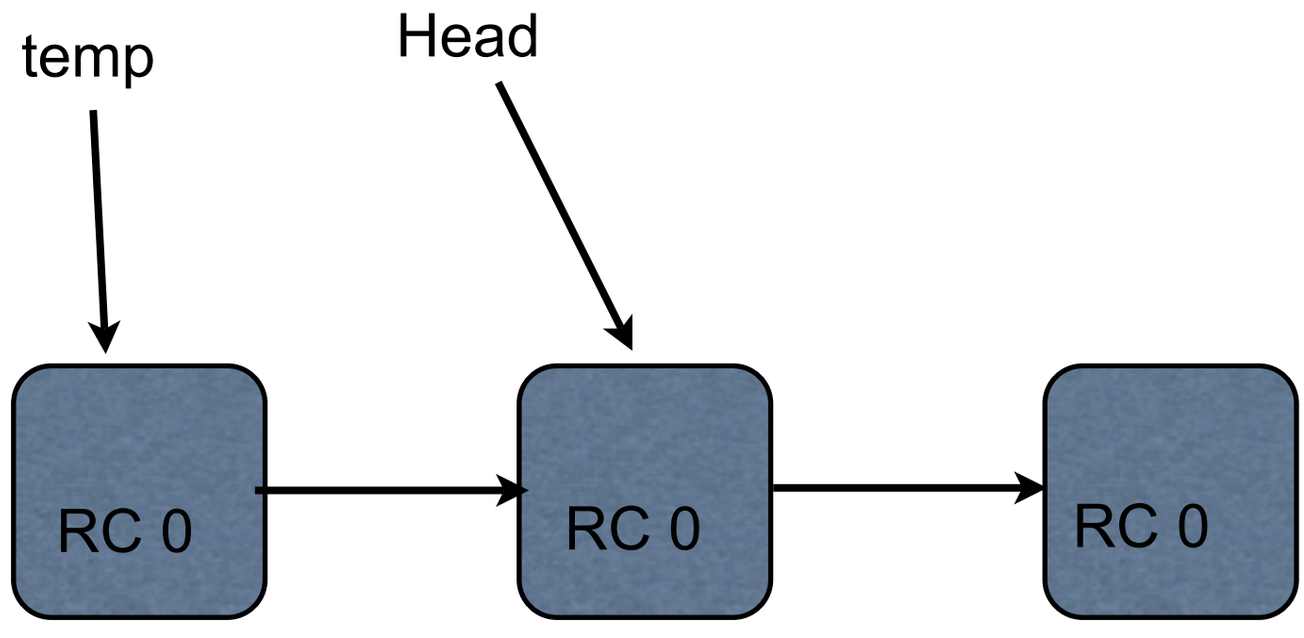
```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        [temp release];
}
```



Set next to nil

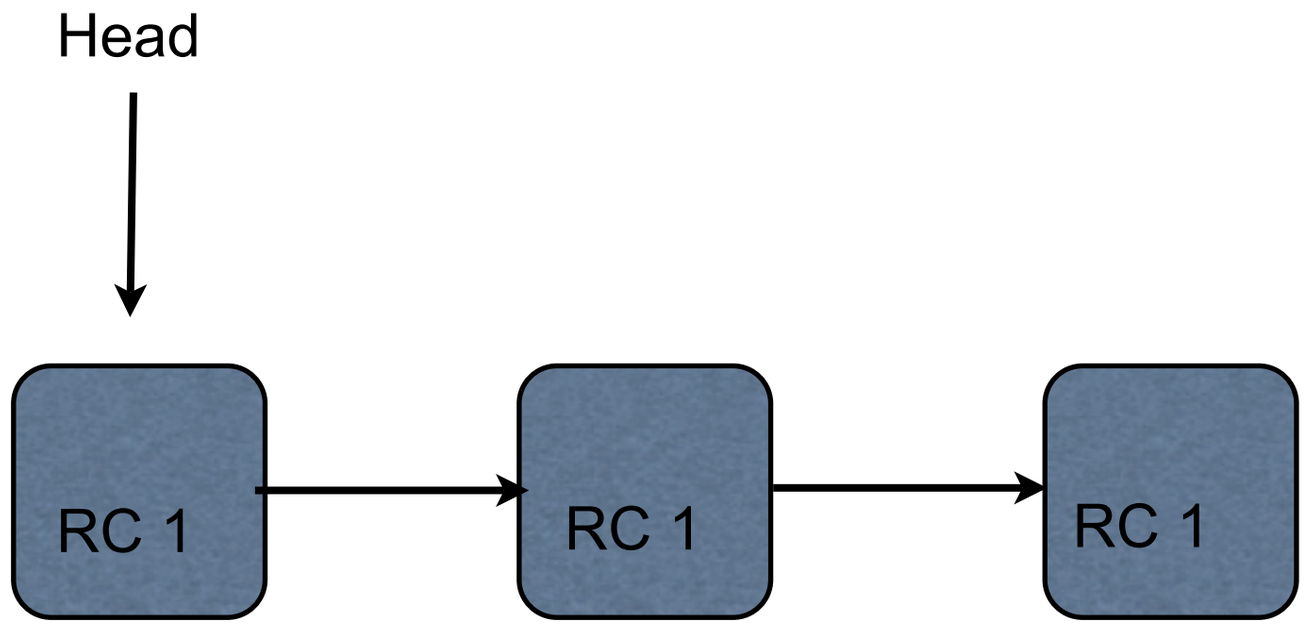
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        temp.next = nil;
    [temp release];
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```



Set next to nil - Slo Mo

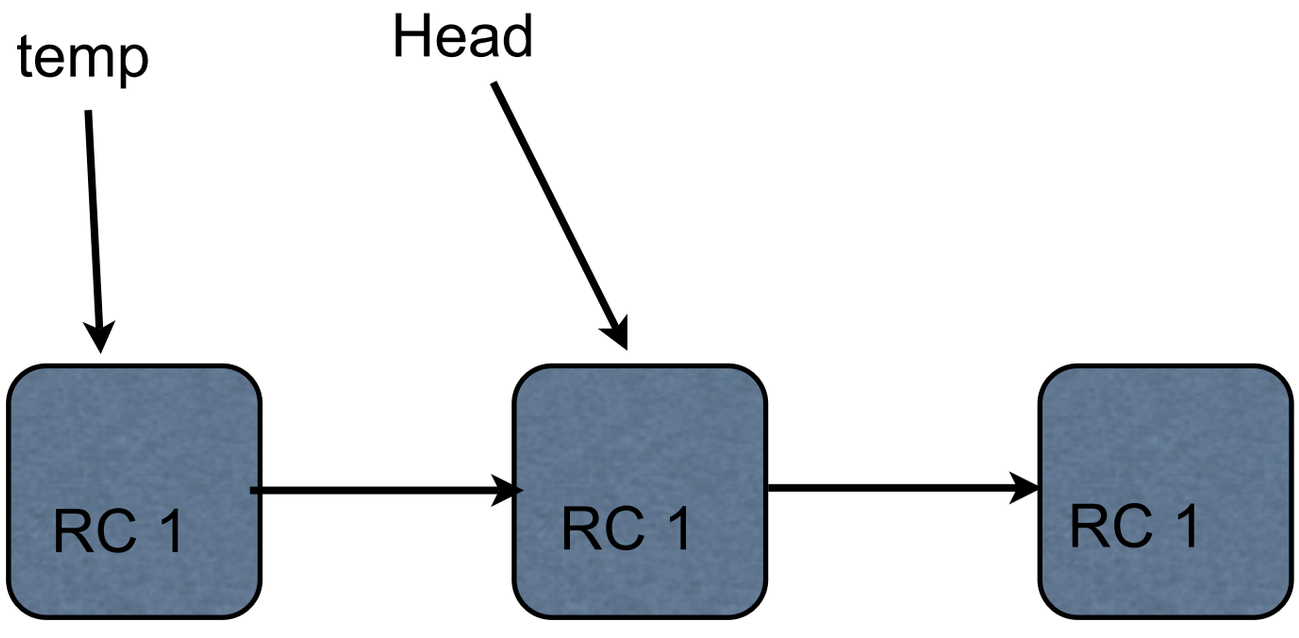
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        temp.next = nil;
    [temp release];
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```



Set next to nil - Slo Mo

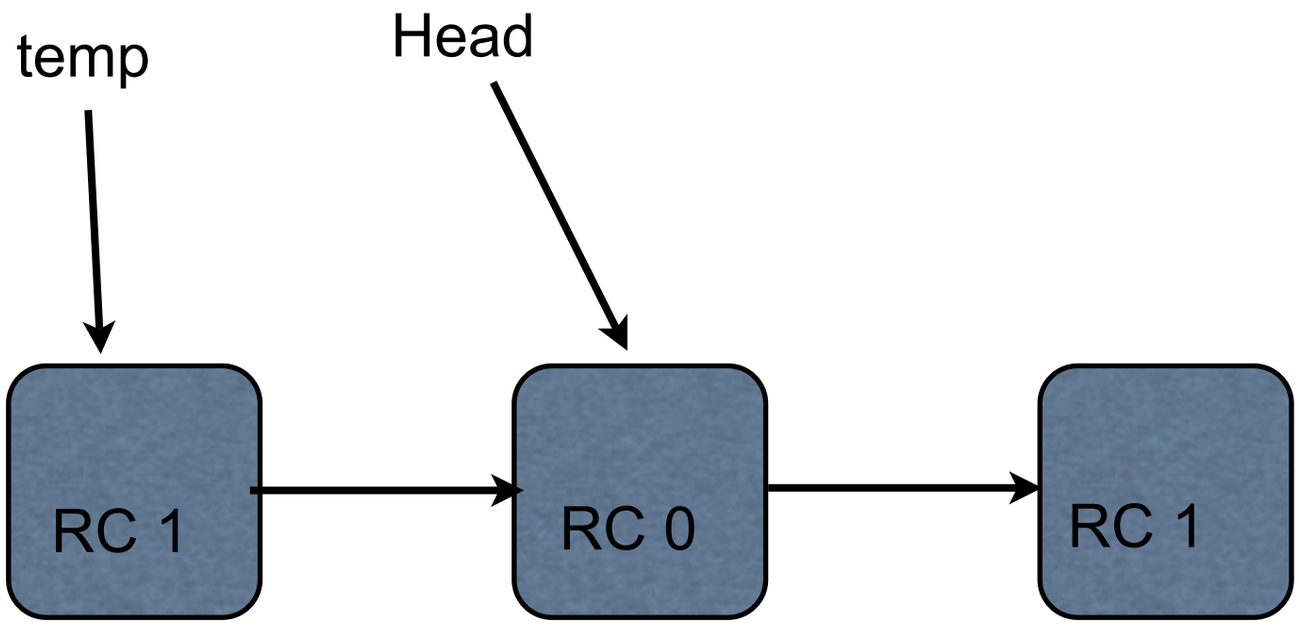
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        temp.next = nil;
    [temp release];
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```



Set next to nil - Slo Mo

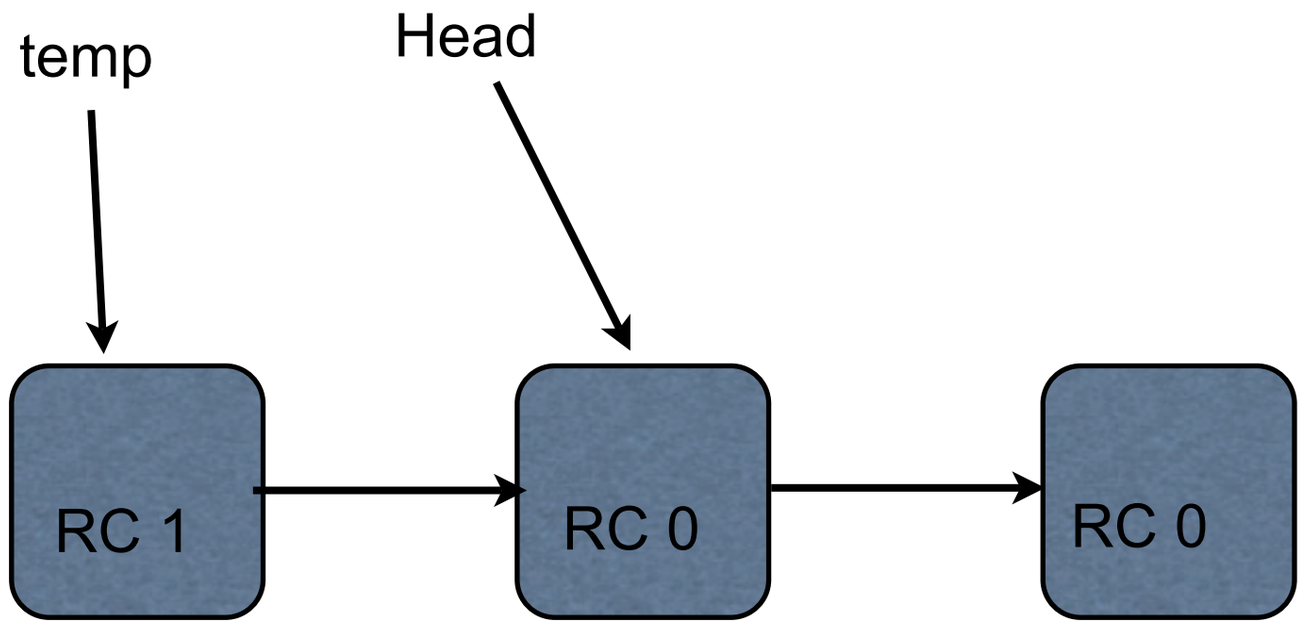
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        temp.next = nil;
    [temp release];
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```



Set next to nil - Slo Mo

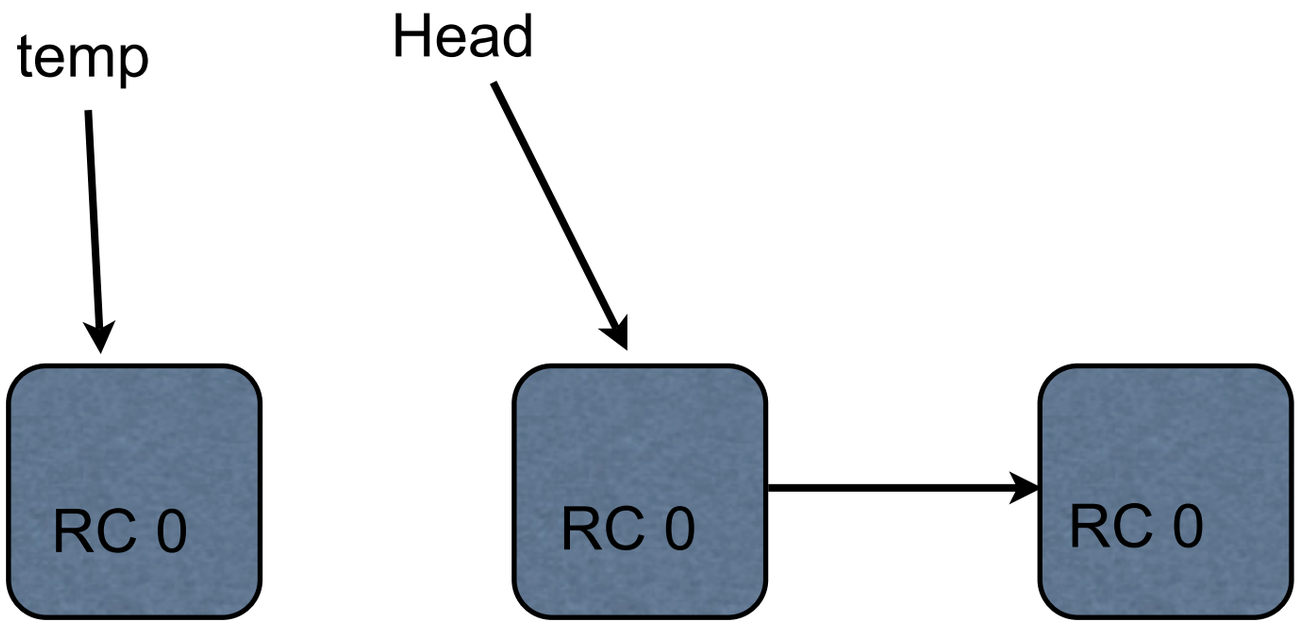
Node

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

LinkedList class

```
- (void) removeFirst{
        Node* temp = head;
        head = head.next;
        count = [NSNumber numberWithInt: ([coun
        temp.next = nil;
    [temp release];
}
```

```
- (void) dealloc{
        [data release];
        [next release];
        [super dealloc];
}
```



Conclusion - Does not work

```
@interface Node : NSObject { }
@property (retain) id data;
@property (retain) Node *next;

@end
```

```
@implementation Node

@synthesize next;
@synthesize data;

- (void) dealloc{
      [data release];
      [next release];
      [super dealloc];
}

@end
```

```
- (void) removeFirst{
      Node* temp = head;
      head = head.next;
      count = [NSNumber numberWithInt: ([coun
      [temp release];
}
```

```
- (void)setNext:(Node *)object {
    if(next != nil) { [next release]; }
    [object retain];
    next = object;
}
```

How to figure this out?

Draw the picture

Step through your code & update picture

```
-(id)initWithArray:(NSArray*)array{
    Linkedlist *arrayList=[[Linkedlist alloc] init];
        for(int i=0;i<[array count] ;i++){
            [arrayList addLast:[array objectAtIndex:i]];
        }

    return arrayList;
}
```

```
-(Node *) addf: (id) item1{
        Node * temp= [[Node alloc] init];
        [temp setitem:item1];
        [temp setlink:self];
        [temp autorelease];
        return temp;
}

-(Node *) addl: (id) item1{

        link = [[Node alloc] init];
        [link setitem: item1];
        [link setlink: nil];
        return link;

}
```

```
[dispIndexElement release];


    if(dispIndexElement.data ==nil)
    {
        NSException *exp=[NSException exceptionWithName:@"IndexOutOfRangeExce
reason:@"Index not in range"  userInfo:nil];
        @throw exp;
    }

    return dispIndexElement.data;
```

```
- (void)enumerateObjectsUsingBlock:(void (^)(id obj, NSUInteger idx, BOOL stop))funct
{
//    NSLog(@"hello!! this is Enum Block");
      Node *temp=[[Node alloc]init];
      temp=first;
      int i=1;
      if(temp.data== nil) { NSLog(@"No Elements to be disply");}
      while(temp.data != nil)
      {
            //NSLog(@"%@",temp.data);
            funct(temp.data,i,true);
            temp=temp.next;
            i++;
      }
      [temp release];
}
```

```
-(id) initlinklist:(id)value
{
        listnode = [[Linkedlistnode alloc]init];

        //creating head node with given value

        listnode->linklistvalue = value;
        listnode->next = NULL;
        head = listnode;
        tail = listnode;
        count = [NSNumber numberWithInt: 1];
        [listnode autorelease];
        return self;
}
```

```
-(void)addFirst:(id)anObject {

    [anObject retain];  //retain object passed in so can create node with it
    Node * newNode = [[Node alloc] initWithObject:anObject];
    [anObject release]; //node owns object so release it
```

```
-(void) removeFirst
{

	// If empty linkedlist.
	if(head ==NULL)
	{
		return;
	}
	else
	{
		Linkedlistnode *tempnode = [[Linkedlistnode alloc]init];
		tempnode = head;
		head = head->next;
		[tempnode release];
		count = [NSNumber numberWithInt: [count intValue] - 1];

	}

}
```

```
-(void) addFirst: (id) addObject
{
	Node *newNode = [[Node alloc] init];
	newNode->data = [addObject copy];
	if (head == nil)
	{
		newNode->next = nil;
		head = newNode;
		count = [NSNumber numberWithInt:1];
		return;
	}
	else {
		newNode->next = head;
		head = newNode;
		count = [NSNumber numberWithInt:[count intValue]+1];
		return;
	}
}
```

```
node *temp =[[node alloc]init];
      temp=head;
```

```
@implementation  NSNumber (nxtCount)
static int nextCount=0;
- (NSNumber*)next {

      NSNumber *n = [NSNumber numberWithInt:[self intValue]+1];
      nextCount++;
      return n;
}

+ (int)callCount {
      [nextCount retain];
      return nextCount;
}
```

```
-(void)addFirst:(id)object{
        [object copy];
        if([[self count]intValue] == 0) {
         [self createFirst:object];          // first node
    } else {
        List_Node *node = [[List_Node alloc] initWithObject:object];
        [node setNext:head];
        int list_count = [[self count] intValue];
              list_count += 1;
              count = [NSNumber numberWithInt:list_count];
          head = node;
              [node autorelease];
              [object autorelease];
    }
}
```

```
node *tempNew =malloc(sizeof(node));
tempNew = firstNode;
```

```
-(id)initWithObject:(id)nodeData:(ListNode*)nextNode{
      if(self = [super init]){
            data =[nodeData retain];
            next= [nextNode retain];
            [nodeData release];
            [nextNode release];
      }
      return self;
}
```

```
- (void)addFirst:(id) objecttoAdd
{
        node *new = malloc(sizeof(node));
        new->next = firstNode ;
        new->info = [objecttoAdd copy];
        firstNode = new ;
        int countnode = [count integerValue]+1;
        count = [NSNumber numberWithInt:countnode];
        [new->info autorelease];


}
```

```
-(id) objectAtIndex: (NSInteger) nodeIndex {
        id temp = headerNode;
        int address = nodeIndex;

        int counterVar = [count integerValue];
        int  i;

        if (address<counterVar && address>0)  {
                for (i=0; i<address; i++) {
                        temp = [temp getLinkToNext];
                }
                [temp autorelease];
                return [temp getElementInfo];
        }
        else {
                [temp autorelease];
        }
}
```


release in atIndex

```
count = [NSNumber numberWithInt:0];



-(void) dealloc
{
	iterator = head;
	while(iterator != NULL)
	{
		free(head);
		head = iterator;
	}

	[count release];
	[super dealloc];
}
```

```
NSString *stringFromFileAtPath=[
    NSString
            stringWithContentsOfFile: filePath
        encoding: NSUTF8StringEncoding
        error:&error];
     ...
     [stringFromFileAtPath release];
```

```
-(void)addFirst:(id) object{
	int listSize = [count intValue];//size of the list

	Node *currentNode = [[Node alloc] initWithObject:object];

	if (self != nil) { //list not null
		[currentNode setNext:head];
		head = currentNode;
	}
	else {
		head = tail = currentNode;
	}
	listSize++;
	count = [NSNumber numberWithInt:listSize];
	[currentNode autorelease];
}
```

```
- (void) removeLast{
        if ([[self count] intValue] == 0) {
                firstNode = nil;
                lastNode = nil;
        }else if ([[self count] intValue] == 1) {
                [firstNode release];
                [lastNode release];
                firstNode = nil;
                lastNode = nil;
                count = [NSNumber numberWithInt:[[self count] intValue]- 1];
                }else {
                Node * iterator = firstNode;
                for( int i = 1; i < ([[self count] intValue] - 1); i++){
                        iterator = [iterator link];
                }
                        [lastNode release];
                        lastNode = iterator;
                        [lastNode retain];
                        [[lastNode link] release];
                        [iterator release];
                count = [NSNumber numberWithInt:[[self count] intValue]- 1];
        }
}
```

```
//linked list                                          see node on next slide
-(void) addFirst:(id)object
{
      size=size+1;
      if([self head] == nil){
            node *firstnodeval=[[node alloc]initlinklist:object:nil];
            head=[firstnodeval retain];
            [firstnodeval autorelease];
      }else{
            node *nextnodeval=[[node alloc]initlinklist:object :[self head]];
            head = [nextnodeval retain];
            [nextnodeval autorelease];
      }
}
```

See Linked List previous slide

```
//node
-(id)initlinklist:(id)nodedata:(node*)nextnodeval{

        if(self = [super init]){
                d = [nodedata retain];
                next = [nextnodeval retain];
                [nodedata autorelease];
                [nextnodeval autorelease];
        }
        return self;
}
```

```
+ (id) new
{
        return [LinkedList alloc];
}
```

```
-(id)initlinklist:(id)nodedata:(node*)nextnodeval{

        if(self = [super init]){
                d = [nodedata retain];
                next = [nextnodeval retain];
                [nodedata autorelease];
                [nextnodeval autorelease];
        }
        return self;
}
```

```
-(void)removeFirst{
        linkedListNode *temp = [[linkedListNode alloc] init];
        temp=first;
        if(temp == nil){
                NSLog(@"List is already empty");
        }else{
                //linkedListNode *second = [[linkedListNode alloc]init];
                first=temp.next;
                NSLog(@"Removed from first data %@",temp.data);
                temp.next=nil;
                temp = nil;
                count= [NSNumber numberWithInt:([count intValue]-1)];
        }
        [temp autorelease];
}
```

init

```
@interface Node : NSObject
{
      id ElementInfo;
      Node *LinkToNext;
}

+(Node *) new;
-(void) setElementInfo: (id) elementData;
-(id) getElementInfo;
-(void) setLinkToNext: (Node *) nextNode;
-(Node *) getLinkToNext;
@end
```

```
@implementation Node

+(id) new{     return  [Node alloc];}

-(void) setElementInfo: (id) elementData{    ElementInfo = elementData;}

-(id) getElementInfo{  return ElementInfo;}

-(void) setLinkToNext: (Node *) nextNode{  LinkToNext = nextNode;}

-(Node *) getLinkToNext{  return LinkToNext;}

- (void) dealloc{
      [LinkToNext release];  //dangerous in removeFirst
      [ElementInfo release];
}

@end
```


Note that alloc with no init, note that link and element are released but not retained. So have to read another class to understand memory management. Note the names. Object implements new, so no need to do it here

```
-(id) init
{
	last=nil;
	first=nil;
	count=[NSNumber numberWithInt:1];
	return self;
}
```

```
+ (id) start{
      return [linkedlist alloc];
}
```

```
front = [local retain];
[local release];
```

```
-(id) initWithArray: (NSArray *) anArray
{
	LinkedListMethods *linkedlst = self;

	int lengthOfArray = [anArray count];
	int i;
	for(i=0; i<lengthOfArray; i++)
	{
		NSString *string = [anArray objectAtIndex:i];
		[linkedlst addFirst:string];
	}
	return linkedlst;
}
```

```
-(id) initWithArray:(NSArray*) anArray {

      SingleLinkedList *singleLinkedList=[SingleLinkedList new];
      for(id element in anArray) {
            [singleLinkedList addLast:element];
      }
      [singleLinkedList autorelease];
      return singleLinkedList;
}
```

```
-(id)initWithArray:(NSArray*) anArray{


      Linkedlist *list=[Linkedlist listWithArray:anArray];
      Node * temp=list->firstnode;
      NSLog(@"**Array elements**");
      while(temp!=nil)
      {
            NSLog(@"\t%@",[temp getitem]);
            temp=[temp getlink];
      }
      return list;

}
```

Types, C, OBJ C

```
BOOL *stopBlock = NO;

BOOL *stop=@"YES";
```

```
typedef struct ListNode ListNode;
struct ListNode {
        id value;
        ListNode *next;
};
```

```
-(void)removeFirst:(id)object
{
        Node *temp = [[Node alloc] initWithObject:object];
        temp=head;
        head=head->link;
        free(temp);
        NSLog(@"deleted");
}
```

C & Objective C types

Problem 2

```
@implementation NumberCount

static int count;

/*- (NSNumber *) next:(int) receiver
{
	count++;
	receiver=receiver+1;
	NSNumber *rec;
	rec=[NSNumber numberWithInt:receiver];
	return rec;
	/*
	count++;
	receiverCount=receiver;
	receiverCount=receiverCount+1;
	return (receiverCount);
}*/
```

```
-(NSNumber *)next
{
      count=count+1;
      return  (NSNumber *)([self integerValue]+1);
}
```

```
@interface NSNumber (NSNumber_next)
	int count;
	-(NSNumber *) next;
	-(int) callCount;
@end
```

Problem 4

```
-(id)objectAtIndex : (NSInteger) index
{
    NSException* error = [NSException exceptionWithName:NSRangeException
reason:@"The index is out of range" userInfo:NULL];
    listnode=head;
    int totalnodes = [count intValue];

    if(index>(totalnodes-1)) @throw error;
```

```
@interface LinkedList : Node <List> {
        Node *headerNode;
}
```

```
+(id)listWithArray:(NSArray *) anArray
{
	LinkedList* ll2 = [[LinkedList alloc] init];

	for(id obj in anArray)
	{
		[ll2 addFirst:obj];
	}

	[ll2 autorelease];
	return ll2;
}
```

```
@interface LinkedList : NSObject <LinkedListProtocol> {

        LinkedListNode * head;
}
-(LinkedListNode *) head;
-(void) setHead: (LinkedListNode *) newLink;
-(void) incrementCount;
-(void) decrementCount;
```

```
-(void) objectAtIndex: (NSInteger) index{
```

```
- (NSString *) description {
        ListNode *tmp;
        tmp = head;
        NSString *str;
        NSMutableString *mstr;
        mstr = [[NSMutableString alloc] initWithCapacity:1000];
        id myObject;
        while (tmp != NULL) {
                myObject= tmp->value;
if([myObject isKindOfClass:[NSNumber class]]) {
        str = [tmp->value stringValue];
}
if([myObject isKindOfClass:[NSString class]]) {
        str = (NSString *)tmp->value;
}
[mstr appendString:str];
[mstr insertString:@"," atIndex:[mstr length]];
tmp = tmp->next;
```

```
-(id) objectAtIndex : (NSInteger) index {
        int i = index;
        id myObject;
        int c = [count intValue]-1;
        if(i > c) {
                @try {
                        NSString    *aFormat;
                        aFormat = [NSString stringWithString:@"The value is %i"];
                        [NSException raise:NSRangeException format:aFormat,i];
                }
                @catch (NSException *e) {
                        if ([[e name] isEqualToString:NSRangeException])
                                NSLog(@"NSRangeException");
                }
        }
```

```
-(void) enumerateObjectsUsingBlock:(void (^)(id, NSUInteger, BOOL ))block{

    Node * temp;
    temp =Start_ptr;

    while (temp!=End_ptr) {

        block (temp, 2, YES);
        temp = [temp next];

        NSLog(@"\nLast value is frobm enumeration");

        }
}
```

```
-(void)removeFirst{
        linkedListNode *temp = [[linkedListNode alloc] init];
        temp=first;
        if(temp == nil){
                NSLog(@"List is already empty");
        }else{
                //linkedListNode *second = [[linkedListNode alloc]init];
                first=temp.next;
                NSLog(@"Removed from first data %@",temp.data);
                temp.next=nil;
                temp = nil;
                count= [NSNumber numberWithInt:([count intValue]-1)];
        }
        [temp autorelease];
}
```

```
-(id)objectAtIndex:(NSInteger)index{
    @try {
        Node *temp=head;
        //int ind=index;
        NSLog(@"index %i",index);
        int y=[count intValue];
//      NSLog(@"%i",y);
        if(index>y) {
            [NSException raise: NSRangeException
                        format: @"Index out of bounds"];
        }
        else {
            for(int i =0; i<index;i++) {
                temp=[temp getLink];
            }
            return [temp getValue];
        }
    }
    @catch(NSException *e {
        NSLog(@"%@",e);
        return NULL;
    }
}
```

```
-(id)objectAtIndex:(NSInteger)index{
        [self getFront];
        for (int i = 0; i < index; i ++){
                [self getNext];
        }
        return [self getCurrent];
}
```

Problem 6

```
    while (new!=nil)
    {
        if(new==firstNode)
        {
            firstString = [NSMutableString stringWithFormat:@"%@%@", tempString,new-
>info];
        }
        else {
            firstString = [NSMutableString stringWithFormat:@"%@,%@", tempString,new-
>info];
        }
        if (new->next==nil) {
            firstString = [NSMutableString stringWithFormat:@"%@,%@", tempString,new-
>info];
        }
        new=new->next;
        tempString = firstString;
    }
```

```
@interface NSNumber (next1)
-(NSNumber*) next;
@end

@interface NSNumber (callcount1)
+(int *)callcount;
@end
```

```
@interface Node : NSObject {

        id Data;
        Node *nextPtr;
}
- (id) getData;
@end
```

```
@interface LinkedList : Node <List>
{
        Node *head;
        NSNumber *count;
}
```

@interface LinkList : Node <List>