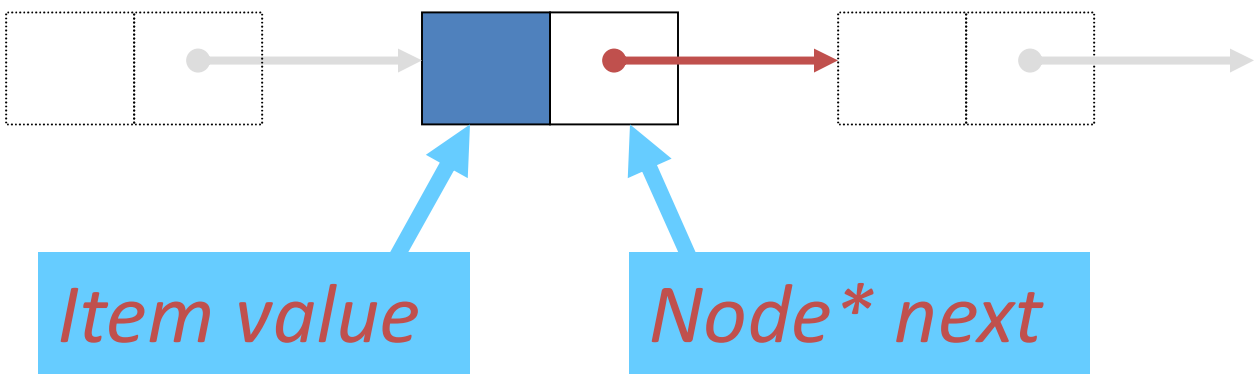
CS 2211
Systems Programming

Part Eleven:

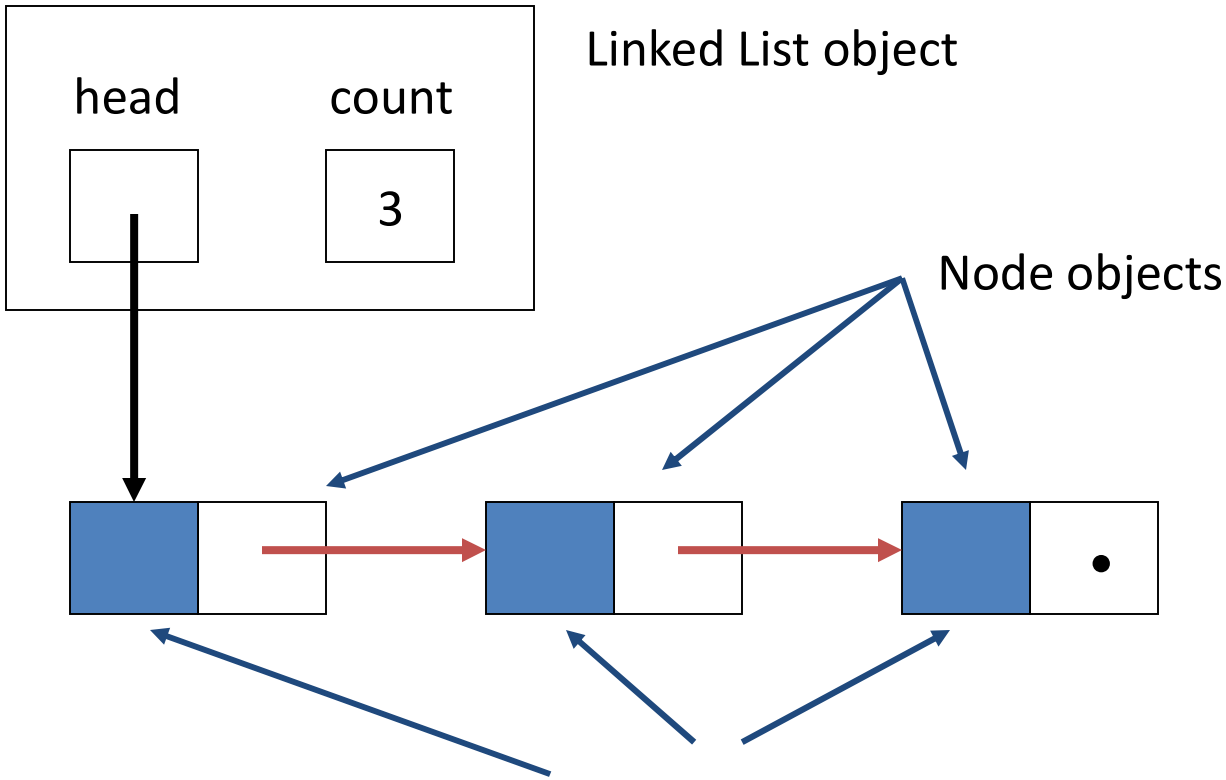
Lists

Nodes in Singly Linked Lists

- Nodes for our linked lists will be objects dynamically created or deleted in the HEAP
  - Each *Node* object in a singly linked list will contain two member variables:



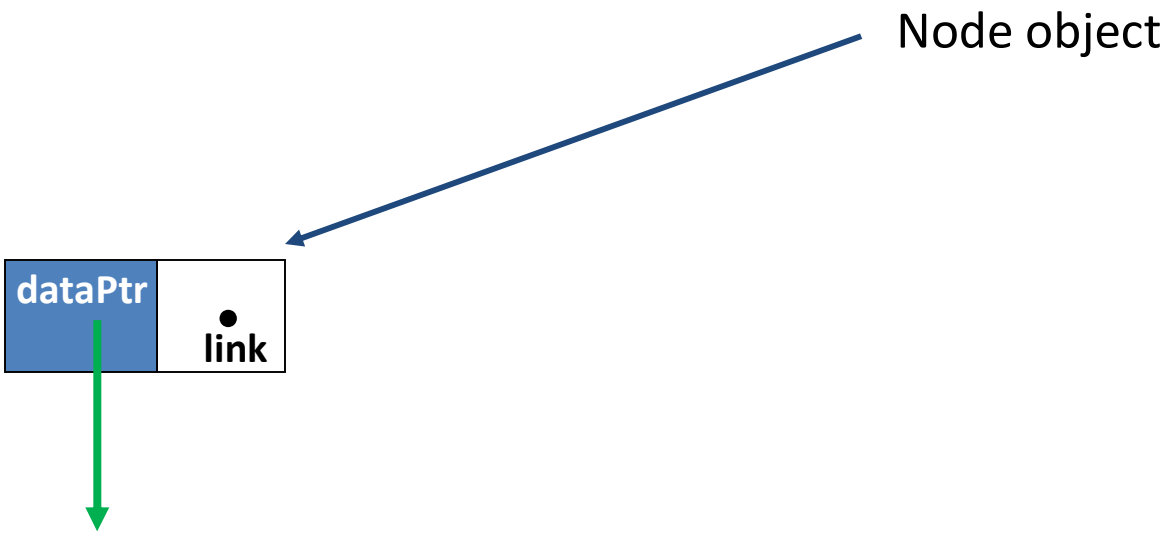
Singly Linked List



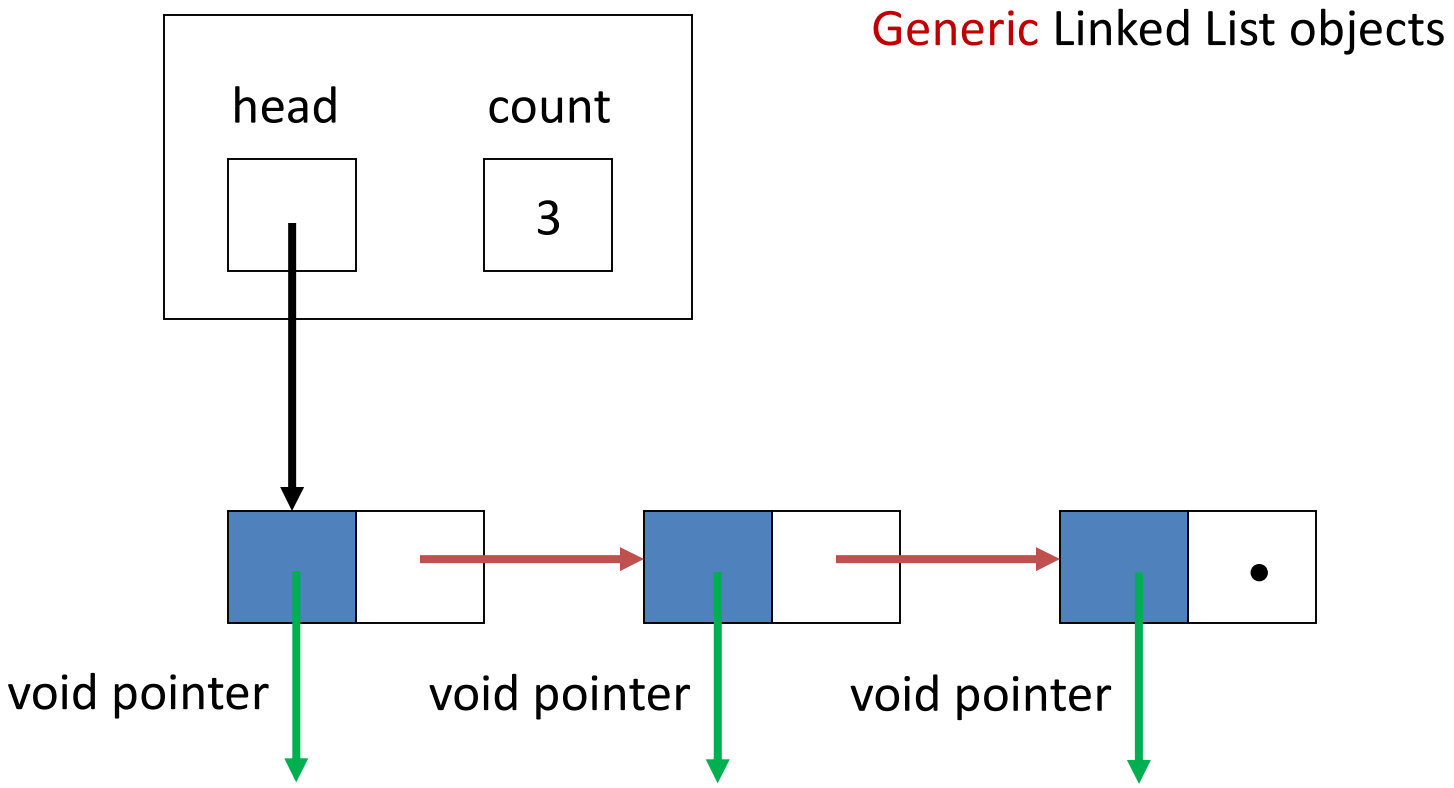
Data objects can be simple data (int, double, etc) or complex data (arrays, structures or unions) or pointers to data outside each individual nodes

```
typedef struct node
{
         void*            dataPtr;
         struct node*     link;
} NODE;

//      ================== createNode ====================

NODE* createNode (void* itemPtr)
{
        NODE* nodePtr;
        nodePtr = (NODE*) malloc (sizeof (NODE));
        nodePtr->dataPtr = itemPtr;
        nodePtr->link    = NULL;
        return nodePtr;
}       // createNode
```

Linked List



Singly Linked List



```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}       // main
```

definitions.h

```
// List ADT Type Defintions
   typedef struct node
   {
      void*          dataPtr;
      struct node*   next;
   } NODE;

   typedef struct
   {
      NODE*    head;
      int      count;
   } LIST;
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}        // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	
	...	
list	510 -513	10100
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}        // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	
	...	
list	510 -513	10100
	...	
	...	
{ DM }	10100 - 10107	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}       // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
              list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	
	...	
list	510 -513	10100
	...	
	...	
head	10100 - 10103	NULL
{ DM }	10104 - 10107	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}       // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	
	...	
list	510 -513	10100
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}       // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}       // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	
	...	
list	510 -513	10100
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

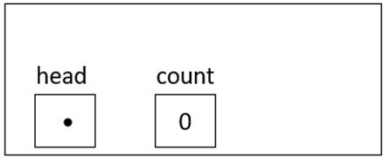
```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}        // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	10100
	...	
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   int*   newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}        // main
```



Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	10100
	...	
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

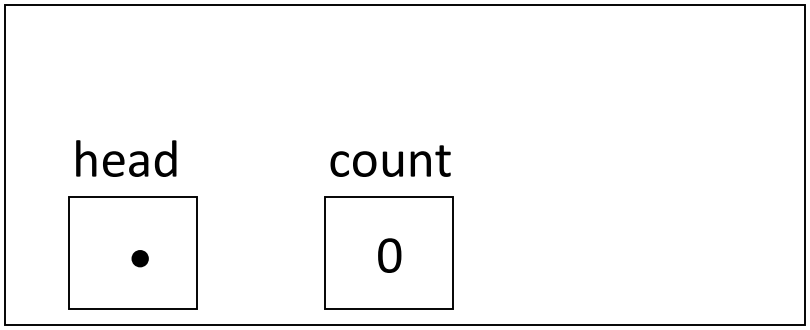
createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

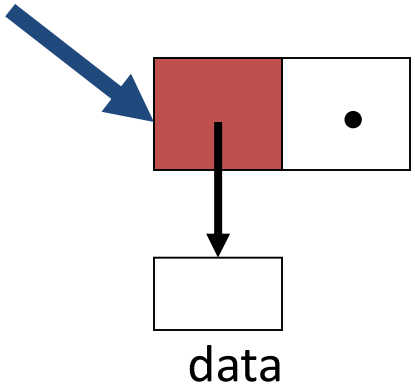
Dynamic Linked Lists in C

END OF PART 1

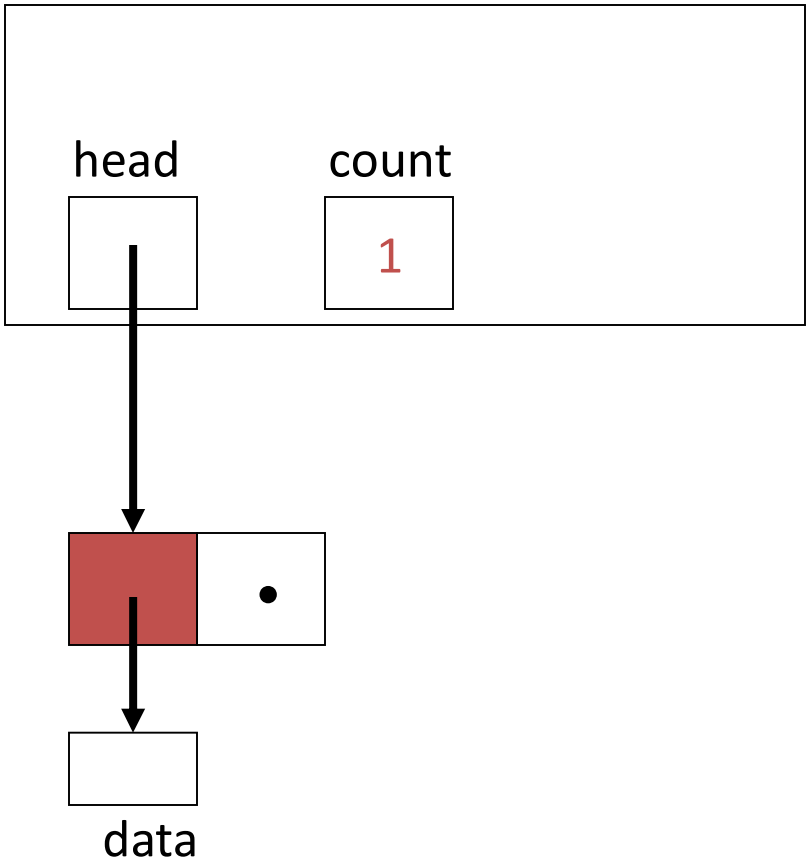
To Add an Item to an Empty Linked List



Build the new node, and put the new data item pointer in it



To Add an Item to an Empty Linked List



Make head point at the new node, and increment the list's size

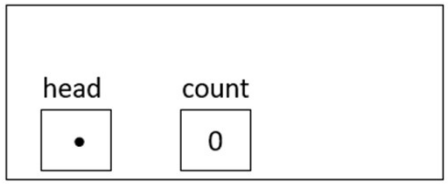
```
#include "mainHeader.h"

int main (void)
{
// Local Definitions
   char* newDataP;
   LIST*  sList;

   sList = createList();
   ...
   return 0;
}        // main
```

createList.c

```
LIST* createList(void)
{
   LIST* list;
   list= (LIST*) malloc (sizeof (LIST));
         if (list)
            {
               list->head = NULL;
               list->count = 0
            } // if
         return list;
}        // createList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	10100
	...	
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 LIST*  sList;

 sList = createList();
   ...

 for (int i = 1; i<=4; i++)
   {
     newDataP = (int*) malloc (sizeof(int));
     *newDataP = i * 3;
     insertList (sList, newDataP);
   }
   ...
}        // main
```



insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

```
    ...
  for (int i = 1; i<=4; i++)
{
    newDataP = (int*) malloc (sizeof(int));
    *newDataP = i * 3;
    insertList(sList, newDataP);
 }
...
}         // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}         // insertList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}         // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}         // insertList
```

Label	Address	Value
newDataP	326 - 329	
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
	...	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
  NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
{ DM }	10400 - 10407	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (! newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	10400
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
{ DM }	10400 - 10407	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	10400
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
{ DM }	10404 - 10407	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	10400
head	10100 - 10103	NULL
count	10104 - 10107	0
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	10400
head	10100 - 10103	NULL
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
I	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	10400
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	10210
newPtr	428 - 431	10400
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

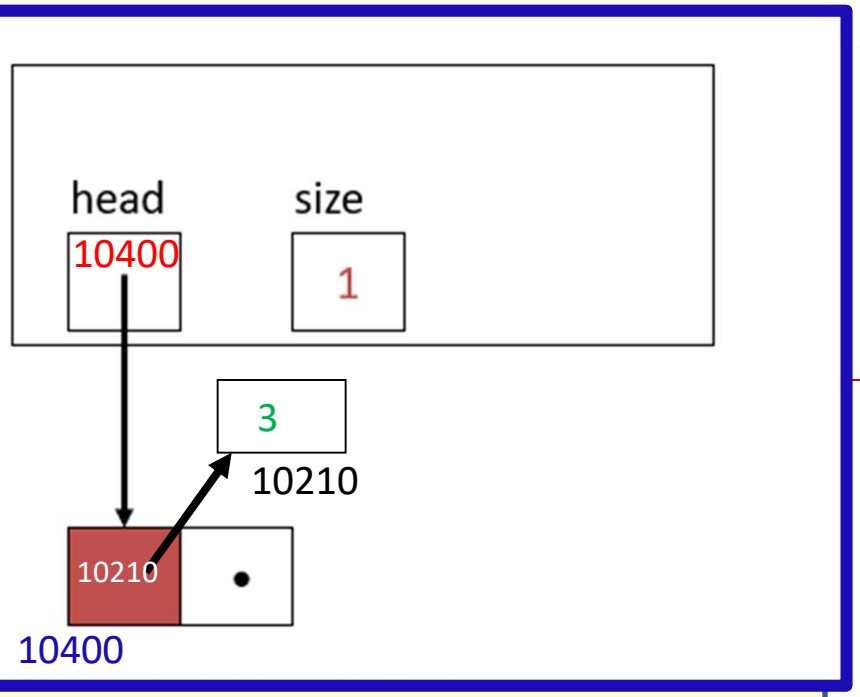
```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
                                                              ;
.
}
```




```
   newPtr->link    = list->head;

   if (list->count == 0)
      list->rear = newPtr;

   (list->count)++;
   list->head = newPtr;
   return true;
}          // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

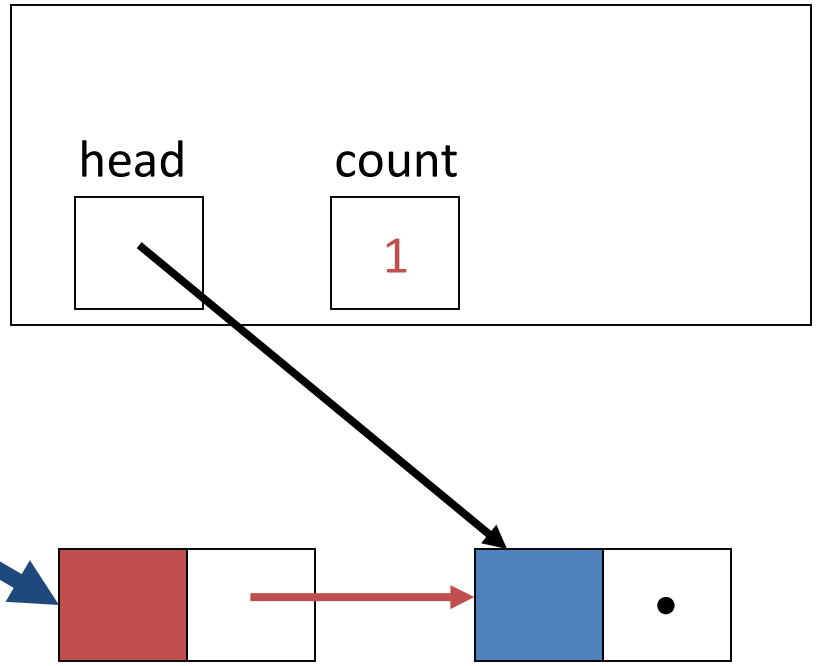
Dynamic Linked Lists in C

END OF PART 2

To Add an Item to the Front of a Linked List

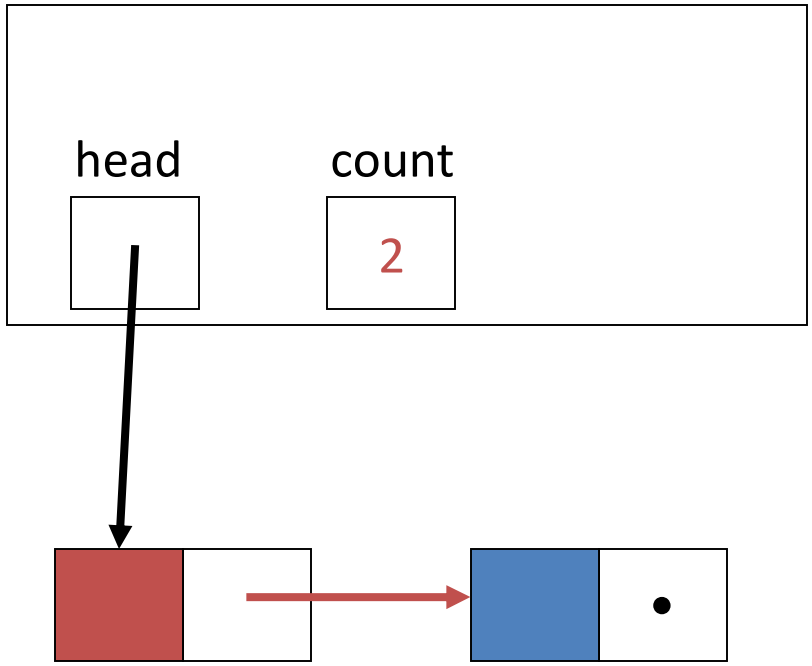
Build the new node, and put the new data item in it,

make new node point to the current front node



To Add an Item to the Front of a Linked List

Reset head so that it points at the new node, and increment size of the linked list



```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	10210
sList	400 - 403	10100
i	404 - 407	2
	...	
	...	
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}         // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}         // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
	...	
	...	
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}         // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}         // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
	...	
	...	
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
	...	
	...	
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
list	420 - 423	10100
itemPtr	424 - 427	12300
	...	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
list	420 - 423	10100
itemPtr	424 - 427	12300
newPtr	428 - 431	
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
       (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
list	420 - 423	10100
itemPtr	424 - 427	12300
newPtr	428 - 431	12560
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
{ DM }	12560 - 12567	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
list	420 - 423	10100
itemPtr	424 - 427	12300
newPtr	428 - 431	12560
head	10100 - 10103	10400
count	10104 - 10107	1
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
list	420 - 423	10100
itemPtr	424 - 427	12300
newPtr	428 - 431	12560
head	10100 - 10103	10400
count	10104 - 10107	2
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
list	420 - 423	10100
itemPtr	424 - 427	12300
newPtr	428 - 431	12560
head	10100 - 10103	12560
count	10104 - 10107	2
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
	...	
	...	
	...	
	...	
	...	
	...	
	...	

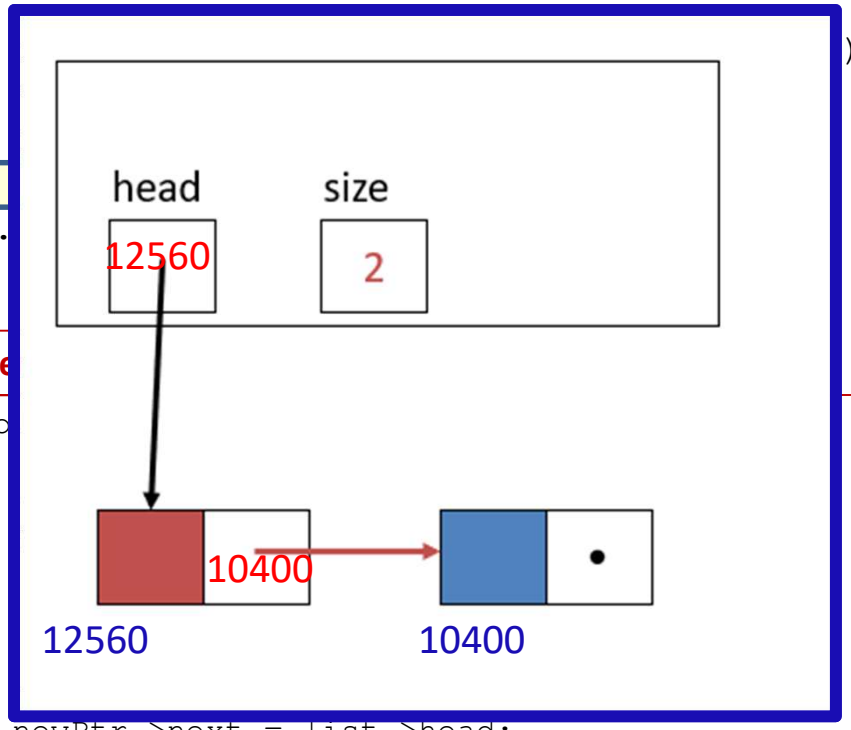
```
    ...
 for (int i = 1; i<=4; i++)
{
    newDataP = (int*) malloc (sizeof(int));
    *newDataP = i * 3;
    insertList(sList, newDataP);
  }
...
}         // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}         // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
	...	
	...	
	...	
head	10100 - 10103	12560
count	10104 - 10107	2
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
	...	
	...	
	...	
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
                                                    );
   }
...
}
```




```
inse
boo
{
    newPtr->next = list->head;

    (list->count)++;
    list->head = newPtr;
    return true;
}          // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	2
	...	
	...	
	...	
head	10100 - 10103	12560
count	10104 - 10107	2
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
	...	
	...	
	...	
	...	
	...	
	...	
	...	

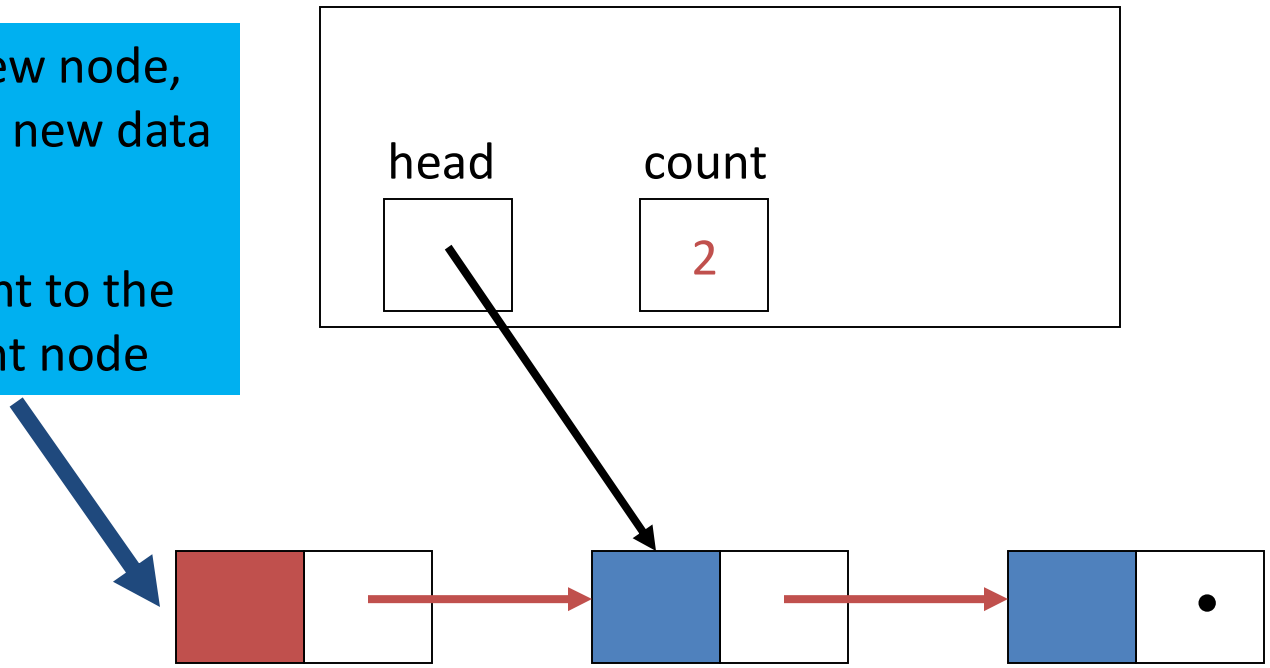
Dynamic Linked Lists in C

END OF PART 3

To Add an Item to the Front of a Linked List

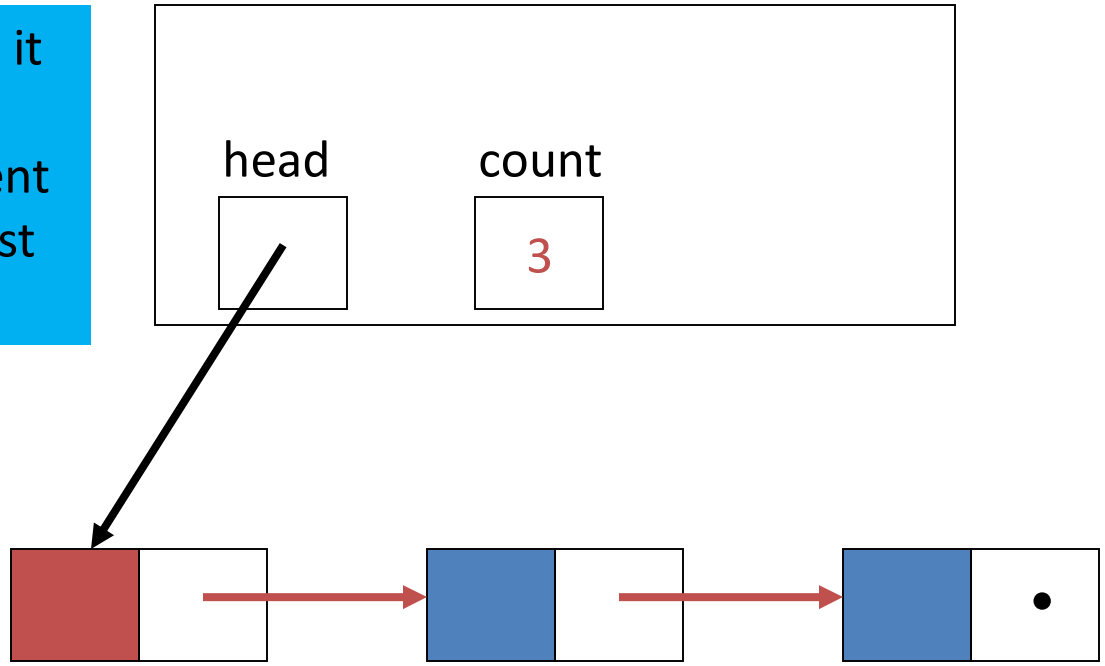
Build the new node, and put the new data item in it,

make it point to the current front node



To Add an Item to the Front of a Linked List

Reset head so that it points at the new node, and increment size of the linked list



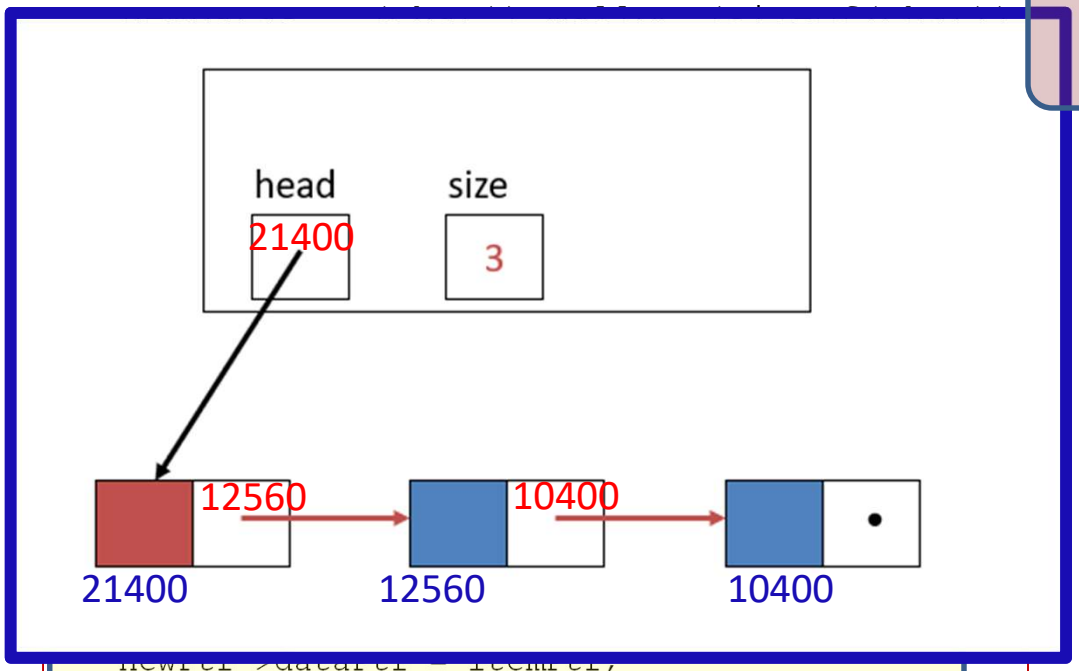
```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
  NODE* newPtr;
  if (!(newPtr =
     (NODE*)malloc(sizeof(NODE))))
    return false;

  newPtr->dataPtr = itemPtr;
  newPtr->next = list->head;

  (list->count)++;
  list->head = newPtr;
  return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	17800
newPtr	428 - 431	21400
head	10100 - 10103	21400
count	10104 - 10107	3
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
	...	
	...	
	...	
	...	

```
   ...
 for (int i = 1; i<=4; i++)
{
```




```
   newPtr->link    = list->head;

   if (list->count == 0)
      list->rear = newPtr;

   (list->count)++;
   list->head = newPtr;
   return true;
}          // insertList
```

Label	Address	Value
newDataP	326 - 329	17800
sList	400 - 403	10100
i	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	17800
newPtr	428 - 431	21400
head	10100 - 10103	21400
count	10104 - 10107	3
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
	...	
	...	
	...	
	...	

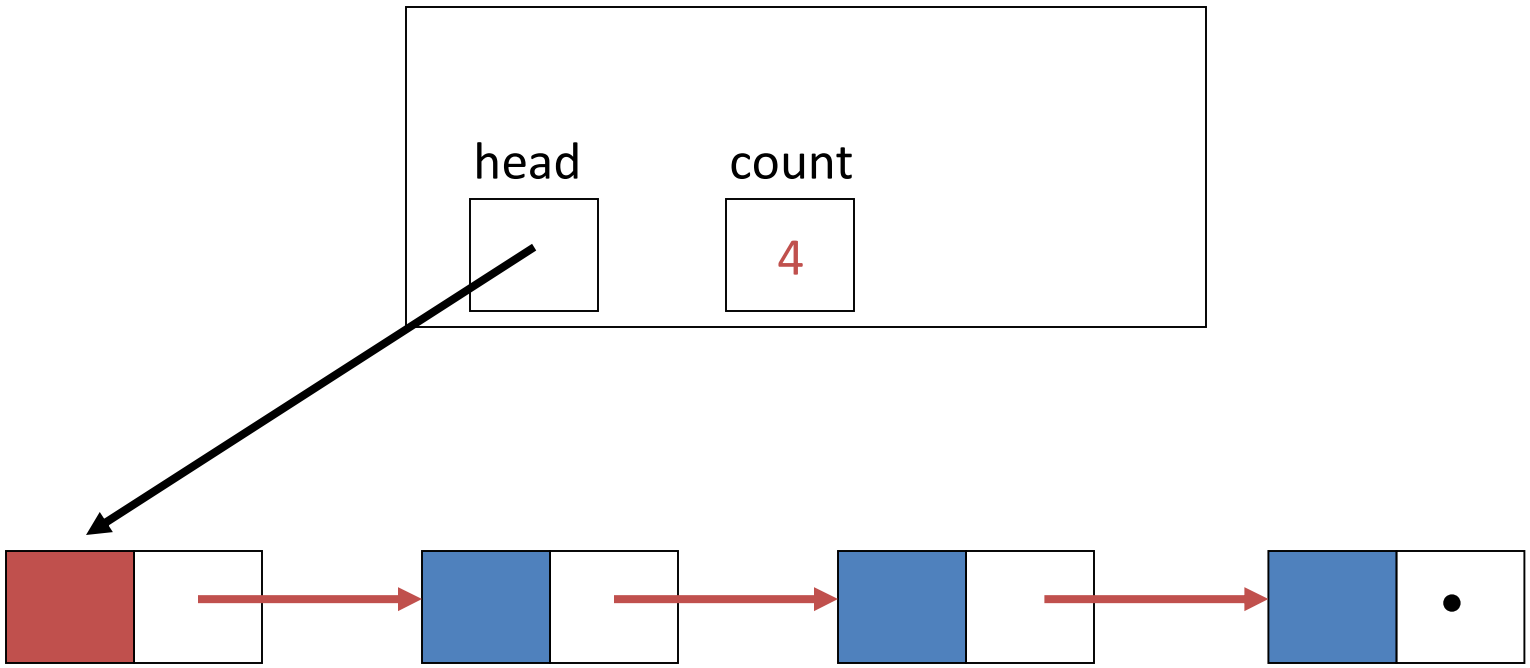
```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}          // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}          // insertList
```

Label	Address	Value
newDataP	326 - 329	17800
sList	400 - 403	10100
	...	
	...	
	...	
	...	
head	10100 - 10103	21400
count	10104 - 10107	3
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
	...	
	...	
	...	
	...	

To Add an Item to the Front of a Linked List



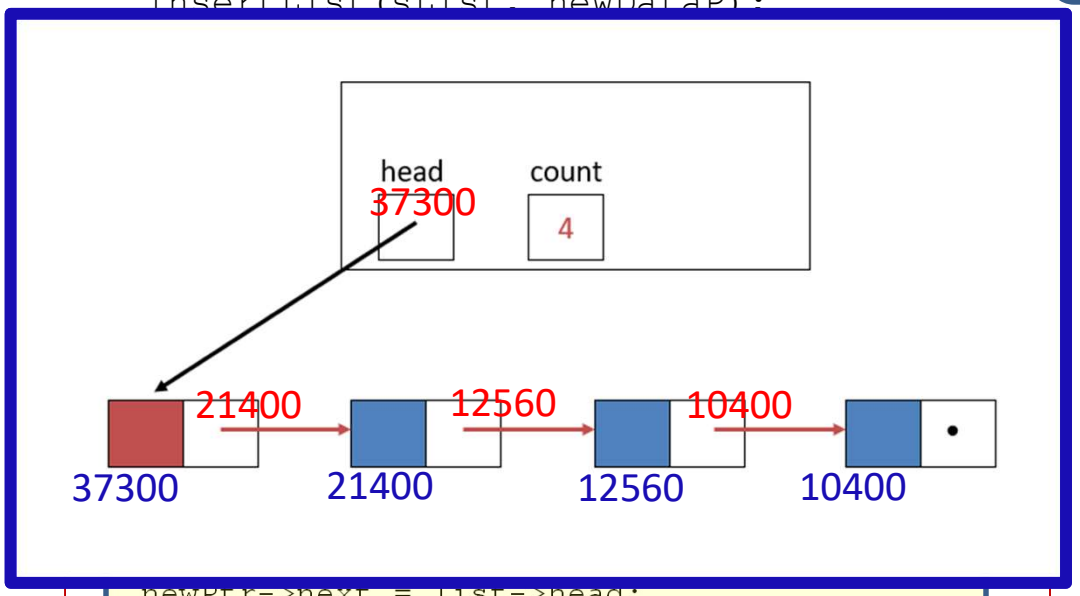
```
   ...
 for (int i = 1; i<=4; i++)
{
   newDataP = (int*) malloc (sizeof(int));
   *newDataP = i * 3;
   insertList(sList, newDataP);
 }
...
}        // main
```

insertList.c

```
bool insertList (LIST* list, void* itemPtr)
{
  NODE* newPtr;
  if (!(newPtr =
     (NODE*)malloc(sizeof(NODE))))
    return false;

  newPtr->dataPtr = itemPtr;
  newPtr->next = list->head;

  (list->count)++;
  list->head = newPtr;
  return true;
}        // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	18300
newPtr	428 - 431	37300
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
    ...
 for (int i = 1; i<=4; i++)
{
    newDataP = (int*) malloc (sizeof(int));
    *newDataP = i * 3;
    insertList(sList, newDataP);
```




```
    newPtr->next = list->head;

    (list->count)++;
    list->head = newPtr;
    return true;
}           // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	1
list	420 - 423	10100
itemPtr	424 - 427	18300
newPtr	428 - 431	37300
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
    ...
  for (int i = 1; i<=4; i++)
{
    newDataP = (int*) malloc (sizeof(int));
    *newDataP = i * 3;
    insertList(sList, newDataP);
  }
...
}          // main
```

insertList.c

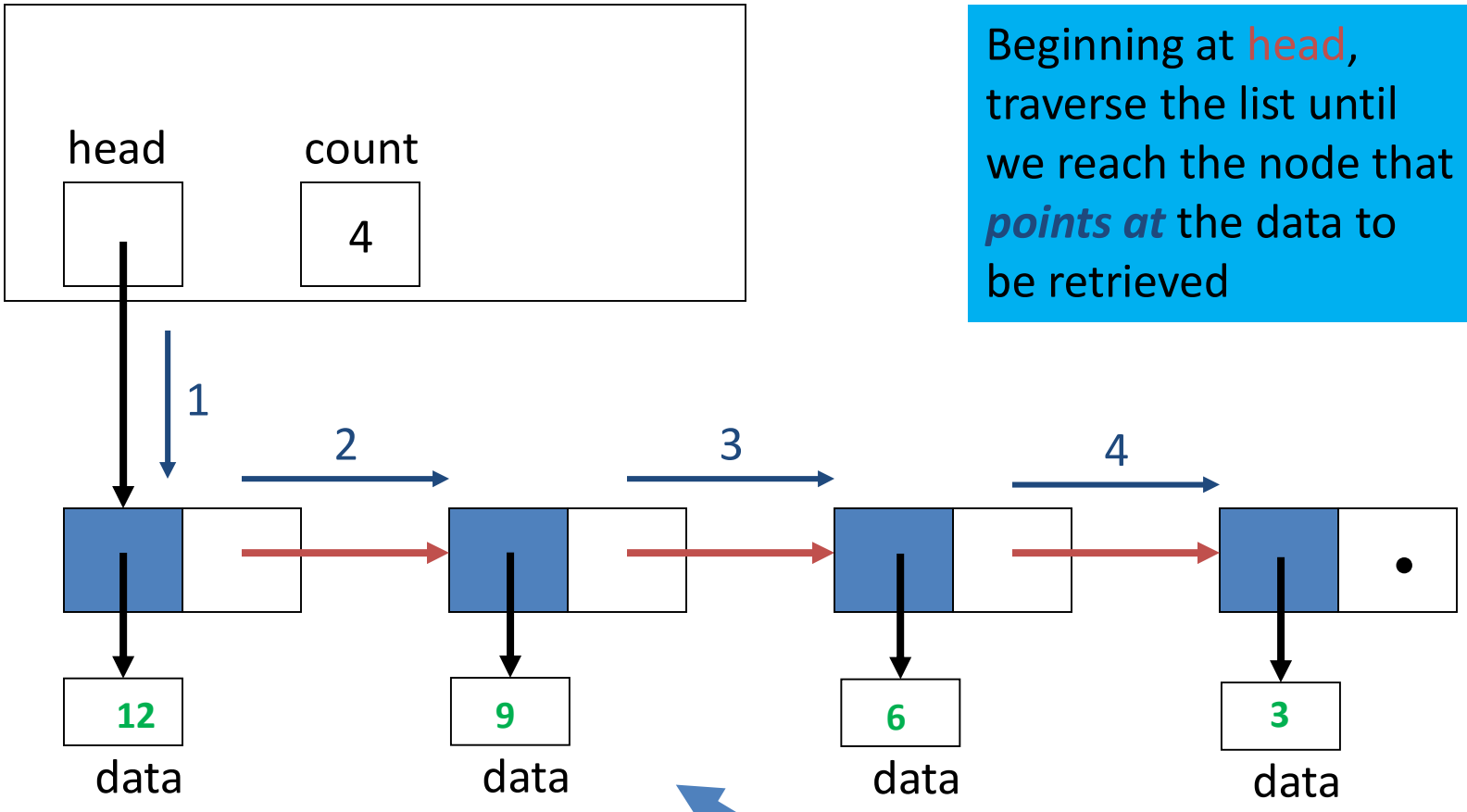
```
bool insertList (LIST* list, void* itemPtr)
{
   NODE* newPtr;
   if (!(newPtr =
      (NODE*)malloc(sizeof(NODE))))
     return false;

   newPtr->dataPtr = itemPtr;
   newPtr->next = list->head;

   (list->count)++;
   list->head = newPtr;
   return true;
}          // insertList
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
i	404 - 407	1
	...	
	...	
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

Dynamic Linked Lists in C

END OF PART 4

Search for an Item in a Linked List



Beginning at head, traverse the list until we reach the node that ***points at*** the data to be retrieved

as you traverse the list
test the value the node points to

```
   ...
   NODE *rValue;
   rValue = searchList(sList, 6);
   if (rValue)
     printf("%d\n",*(int *)rValue->dataPtr;);
...
}          // main
```

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
	...	
	...	
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
	…	
	…	
	…	
head	10100 - 10103	37300
count	10104 - 10107	4
	…	
	…	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	…	

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
List	520 - 523	10100
value	524 - 527	6
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	37300
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	37300
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
            return nP;
   return NULL;
}
```

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	21400
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	21400
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
            return nP;
   return NULL;
}
```

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	12560
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	21400
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
            return nP;
   return NULL;
}
```

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}        // main
```

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
            return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	
list	520 - 523	10100
value	524 - 527	6
nP	528 - 531	21400
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
    ...
  NODE *rValue;
  rValue = searchList(sList, 6);
  if (rValue)
    printf("%d\n",*(int *)rValue->dataPtr;);
...
}         // main
```

searchList.c

```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	21400
	…	
	…	
	…	
head	10100 - 10103	37300
count	10104 - 10107	4
	…	
	…	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	…	

```
   ...
 NODE *rValue;
 rValue = searchList(sList, 6);
 if (rValue)
   printf("%d\n",*(int *)rValue->dataPtr;);
...
}         // main
```

searchList.c

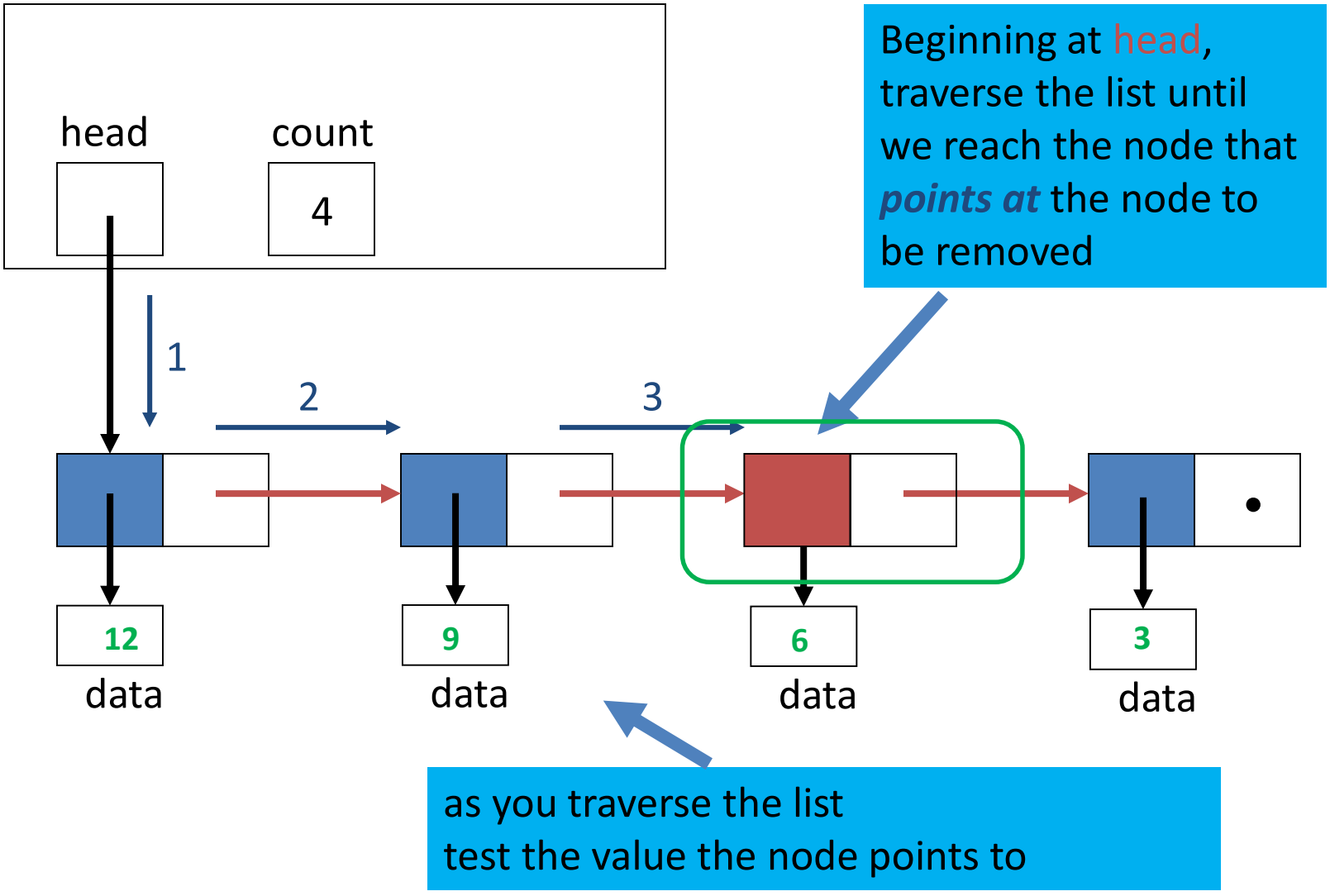
```
NODE* searchList(LIST* list, int value)
{
   NODE *nP;

   for (nP = list->head; nP != NULL; nP = nP->next)
        if (*(int*)nP->dataPtr == value)
           return nP;
   return NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
rValue	404 - 407	21400
	…	
	…	
	…	
head	10100 - 10103	37300
count	10104 - 10107	4
	…	
	…	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	…	

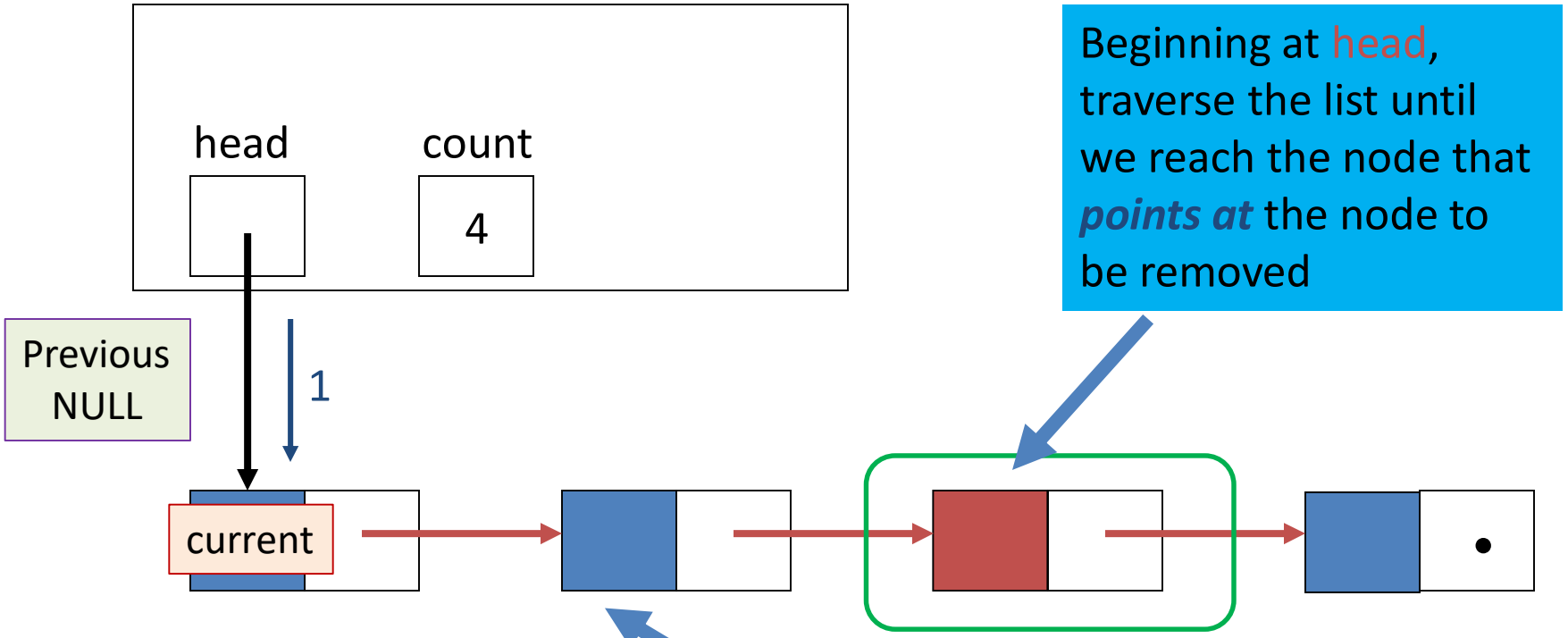
Dynamic Linked Lists in C

END OF PART 5

To Remove an Item From a Linked List

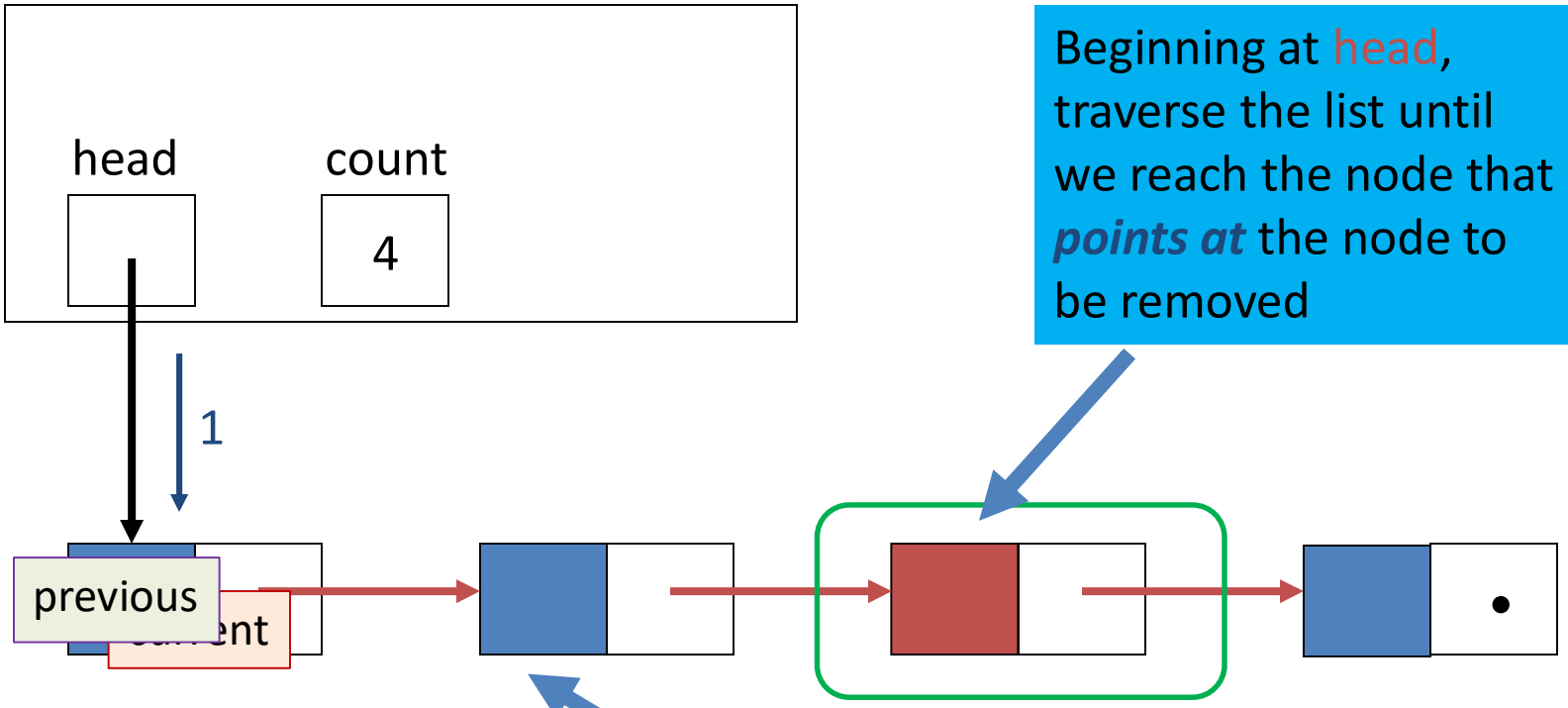


To Remove an Item From a Linked List



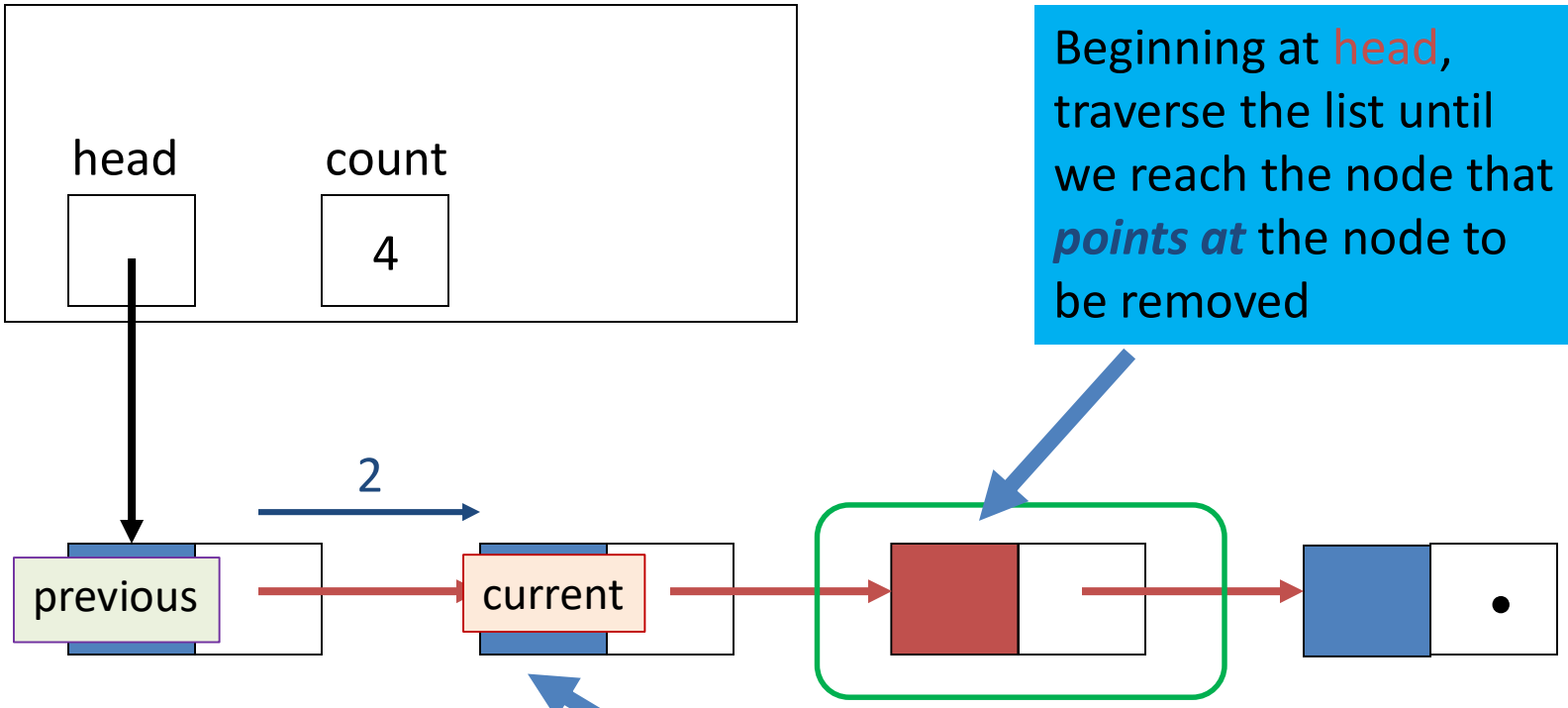

```
cur = list->head;
<loop>
( if not cur->data == value)
      prev = cur;
      cur = cur->next
```

To Remove an Item From a Linked List




```
cur = list->head;
<loop>
( if not cur->data == value)
      prev = cur;
      cur = cur->next
```

To Remove an Item From a Linked List

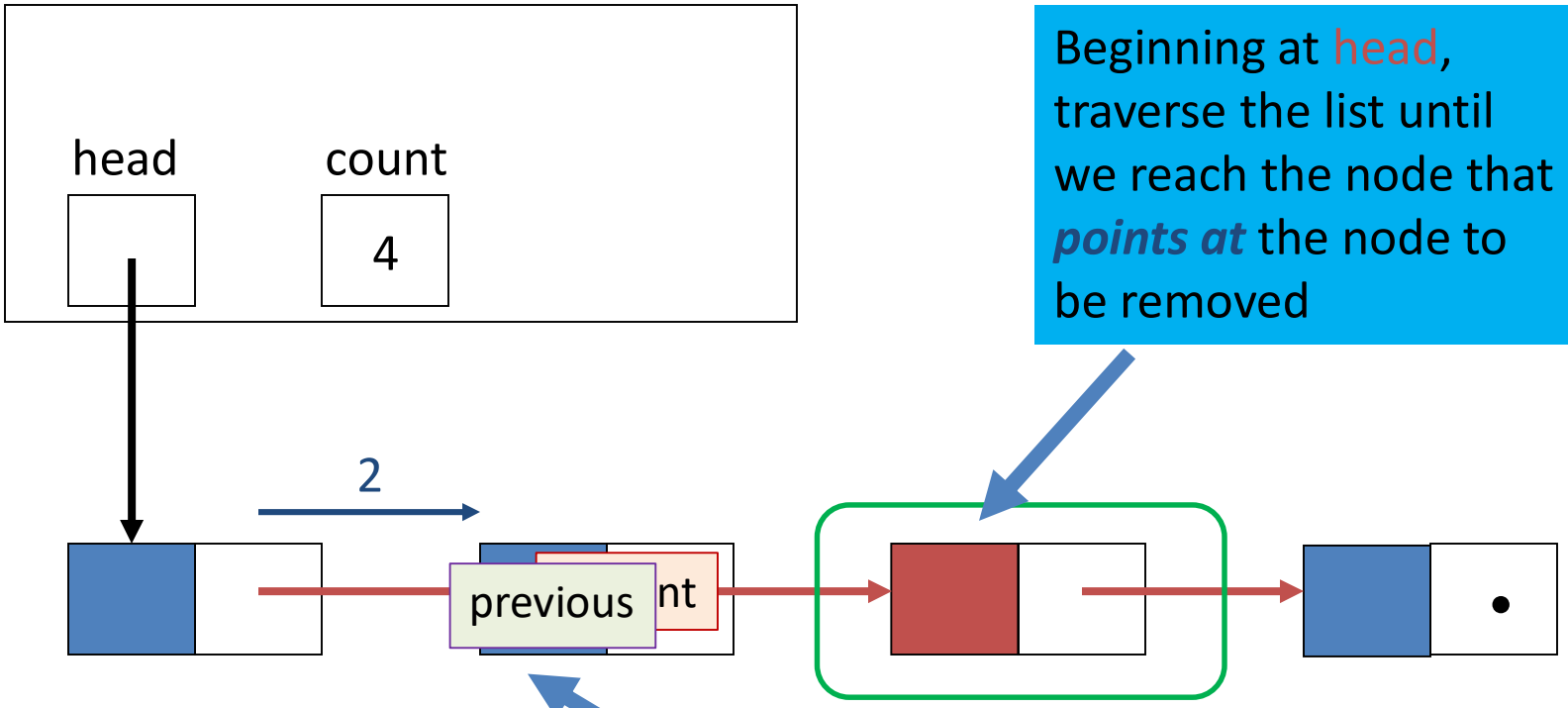


Beginning at head, traverse the list until we reach the node that ***points at*** the node to be removed

```
cur = list->head;
<loop>
( if not cur->data == value)
      prev = cur;
      cur = cur->next
```

as you traverse the list
store the address of the previous node in a temporary variable

To Remove an Item From a Linked List

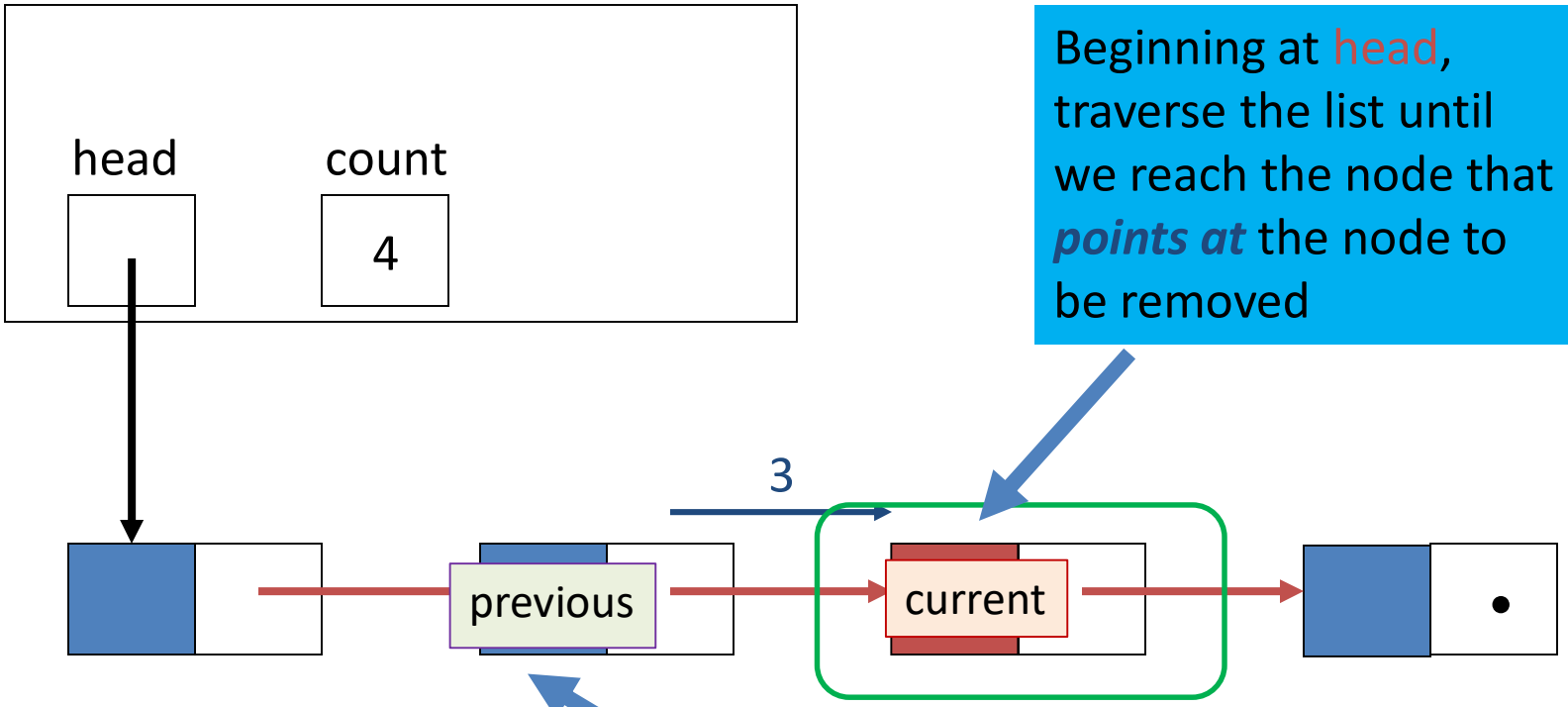


Beginning at head, traverse the list until we reach the node that ***points at*** the node to be removed

```
cur = list->head;
<loop>
( if not cur->data == value)
      prev = cur;
      cur = cur->next
```

as you traverse the list
store the address of the previous node in a temporary variable

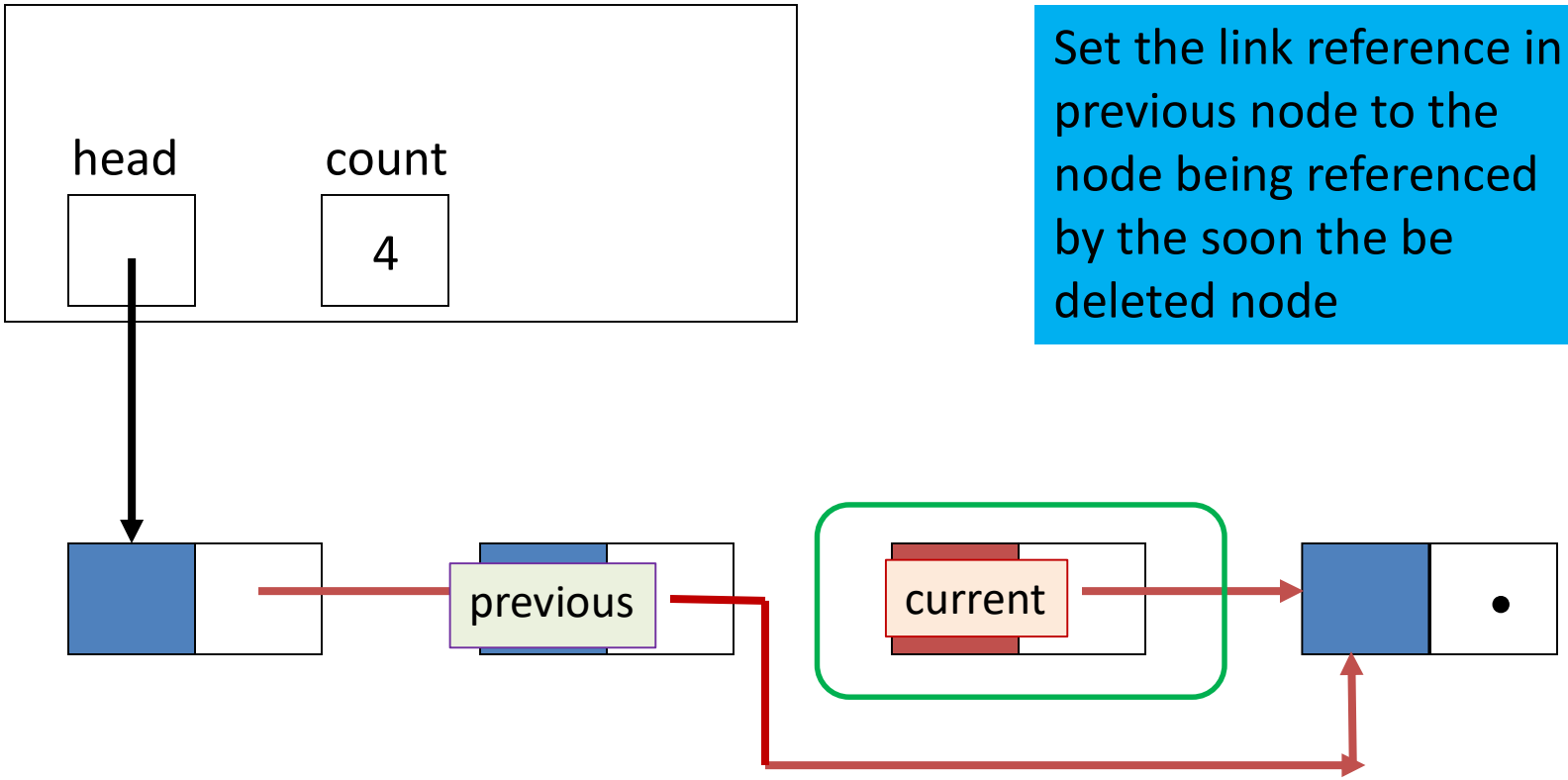
To Remove an Item From a Linked List




```
cur = list->head;
<loop>
( if not cur->data == value)
      prev = cur;
      cur = cur->next
```

as you traverse the list
store the address of the previous node in a temporary variable

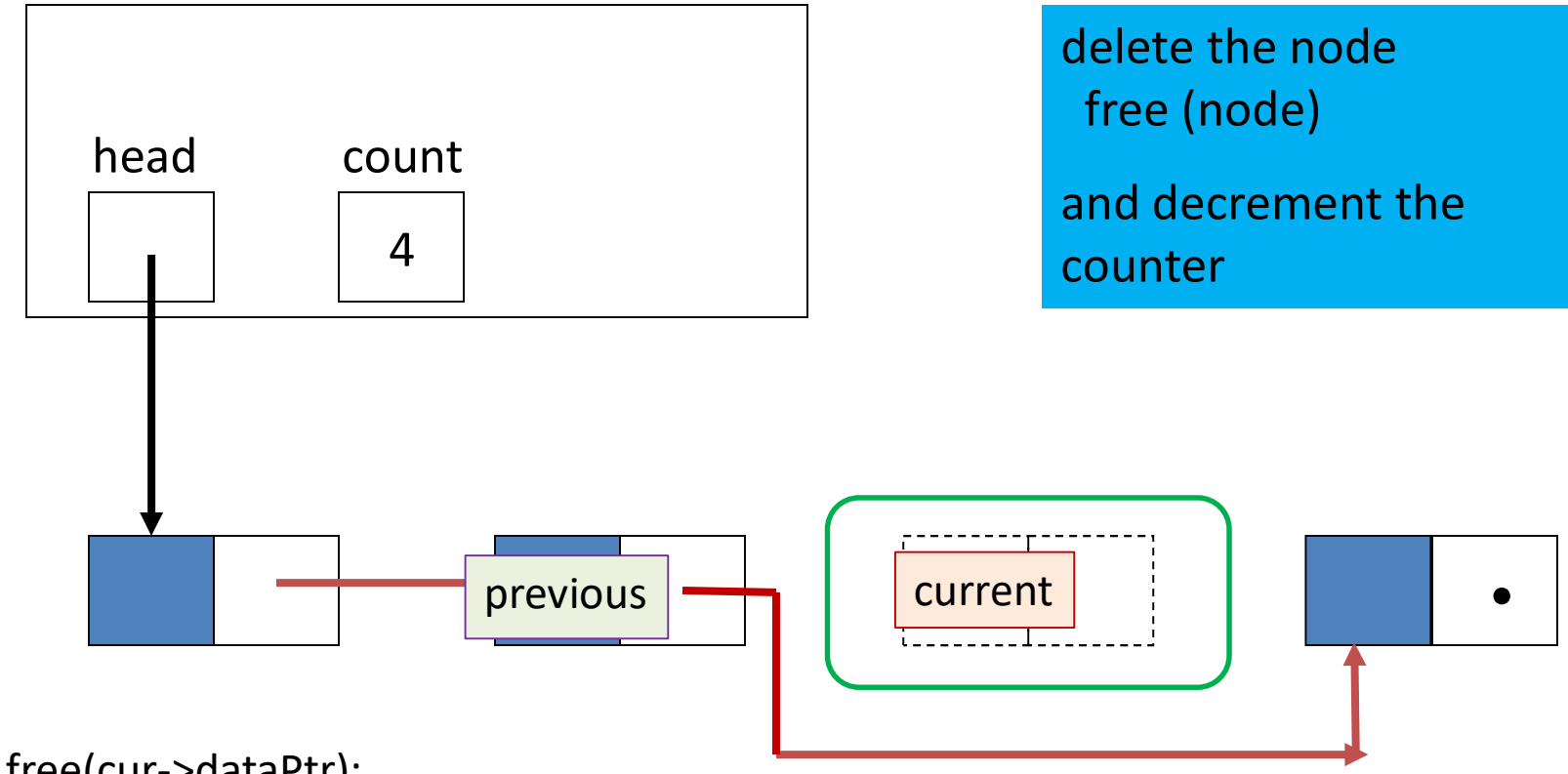
To Remove an Item From a Linked List



Set the link reference in previous node to the node being referenced by the soon the be deleted node

prev->next = cur->next;

To Remove an Item From a Linked List

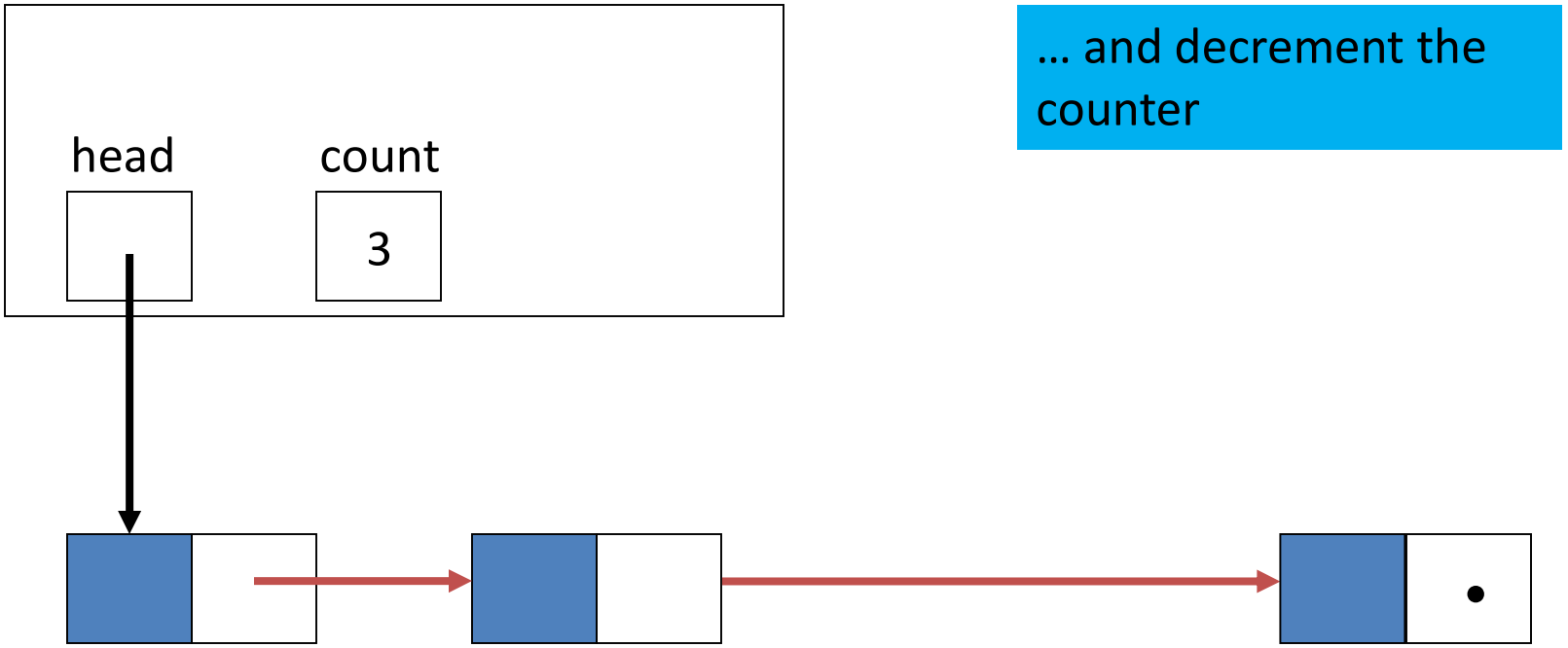


delete the node
free (node)

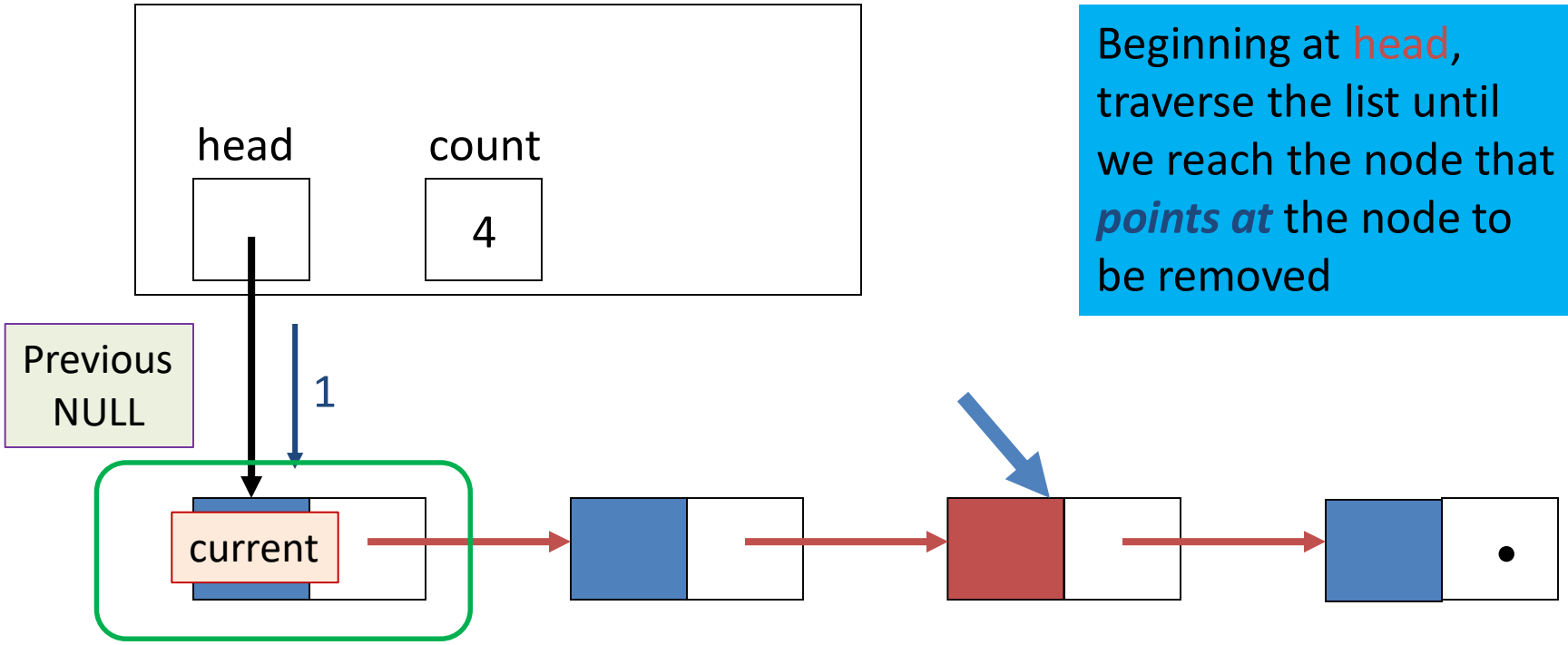
and decrement the counter

```
free(cur->dataPtr);
free(cur);
```

To Remove an Item From a Linked List



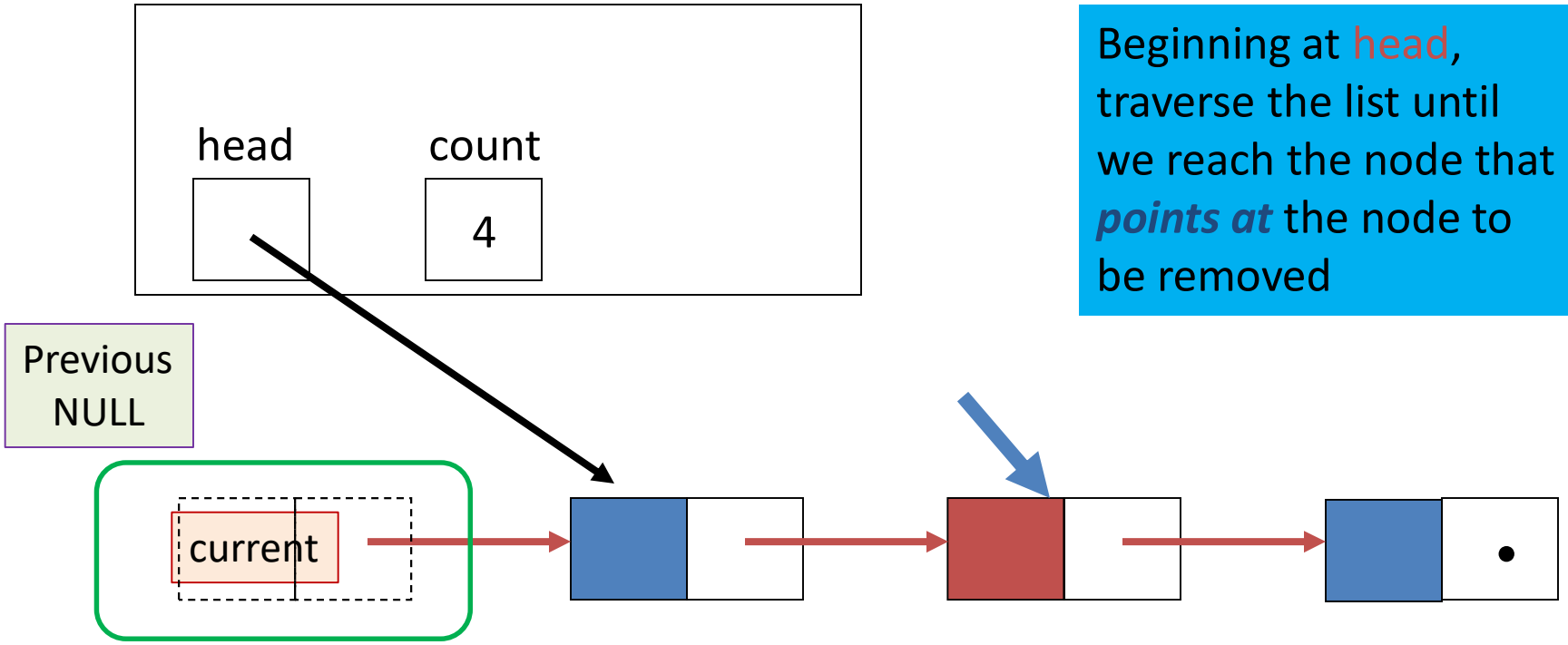
IF Node is First Item From a Linked List



Beginning at head, traverse the list until we reach the node that ***points at*** the node to be removed

```
cur = list->head;
<loop>
( if not cur->data == value)
      prev = cur;
      cur = cur->next
```

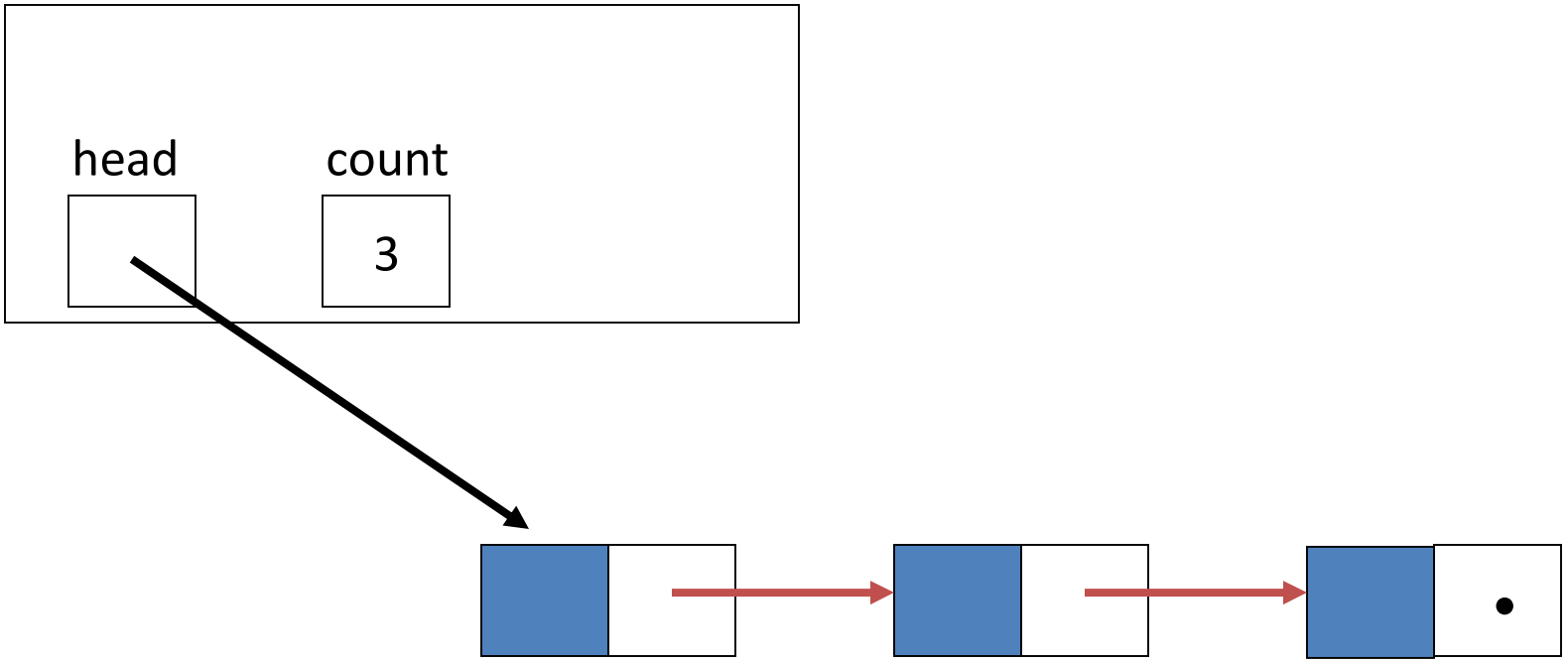
IF Node is First Item From a Linked List



Beginning at head, traverse the list until we reach the node that ***points at*** the node to be removed

```
list->head = cur->next;

free(cur->dataPtr);
free(cur);
```

To Remove an Item From a Linked List

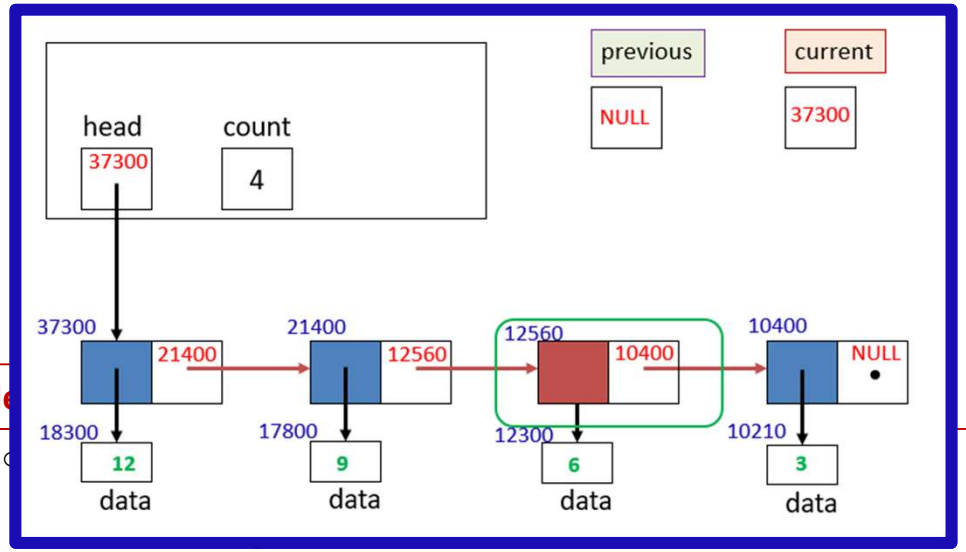


```
    ...
    deleteValue(sList, 6);
    ...
}          // main
```

deleteValue.c

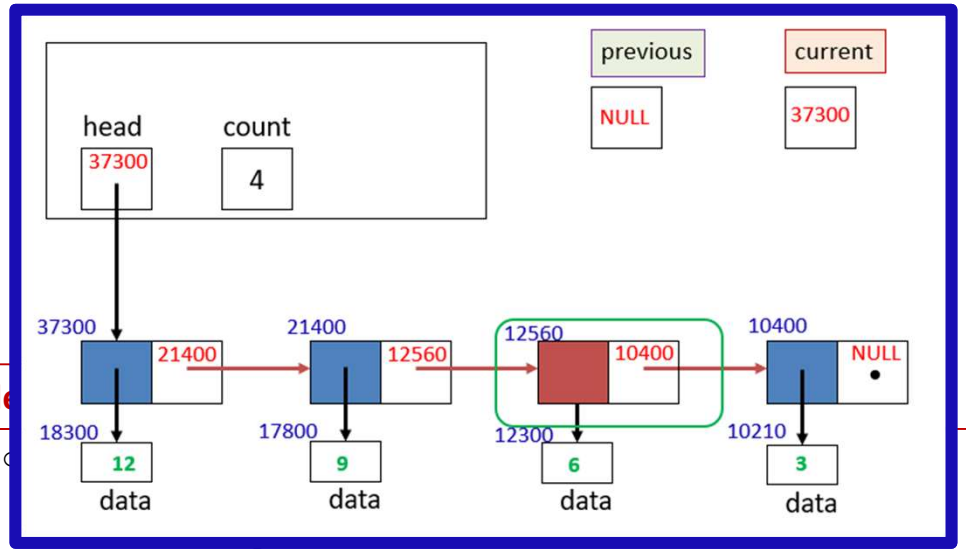
```
bool deleteValue(LIST* list, int value)
{
  NODE *cur, *prev;

  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;              /* n was not found */
  if (prev == NULL)
    list->head = cur->next;   /*n is the first node*/
  else
    prev->next = cur->next;   /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	37300
prev	564 - 567	NULL
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



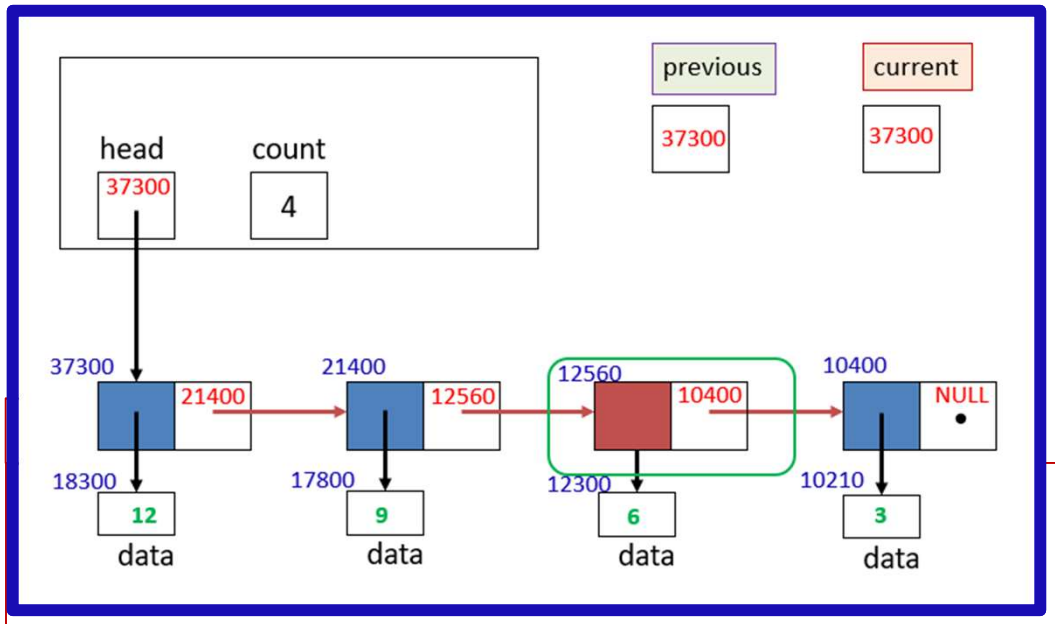

```
de
bo
{
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	37300
prev	564 - 567	NULL
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



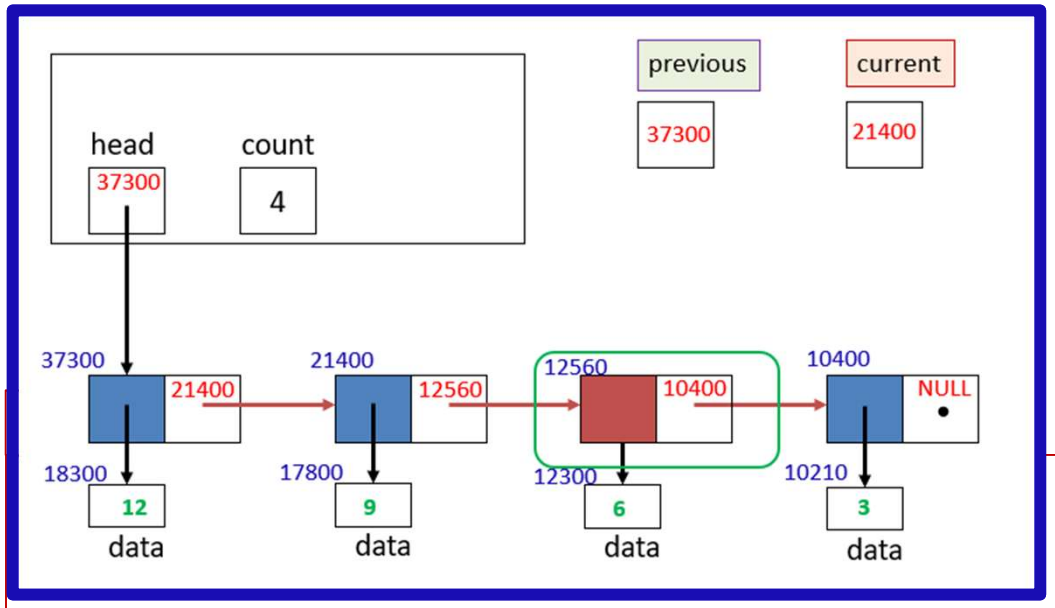

```
de
bo
{

  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	37300
prev	564 - 567	NULL
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



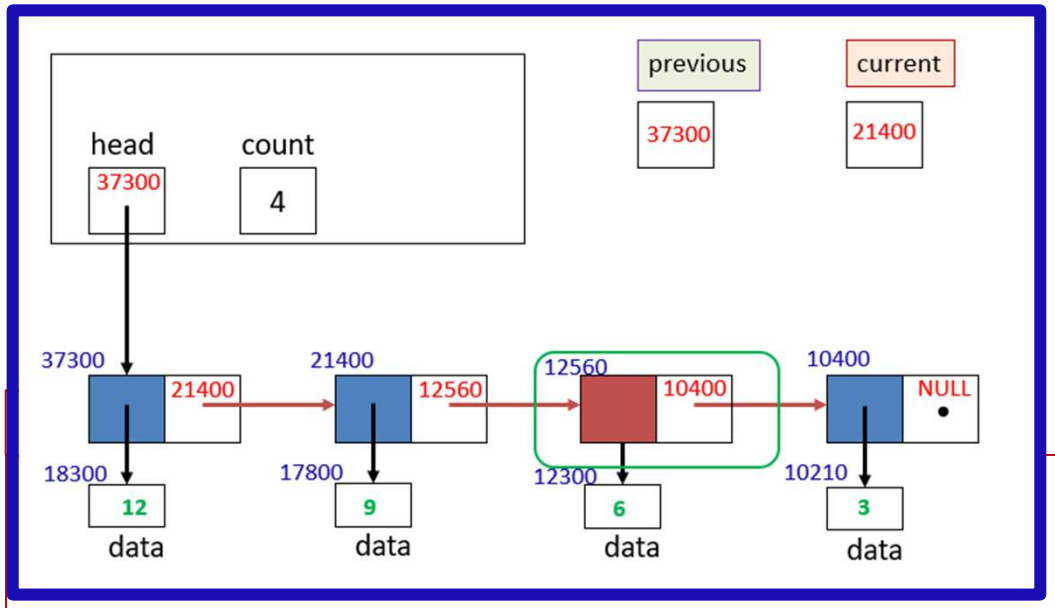

```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;   /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	37300
prev	564 - 567	37300
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



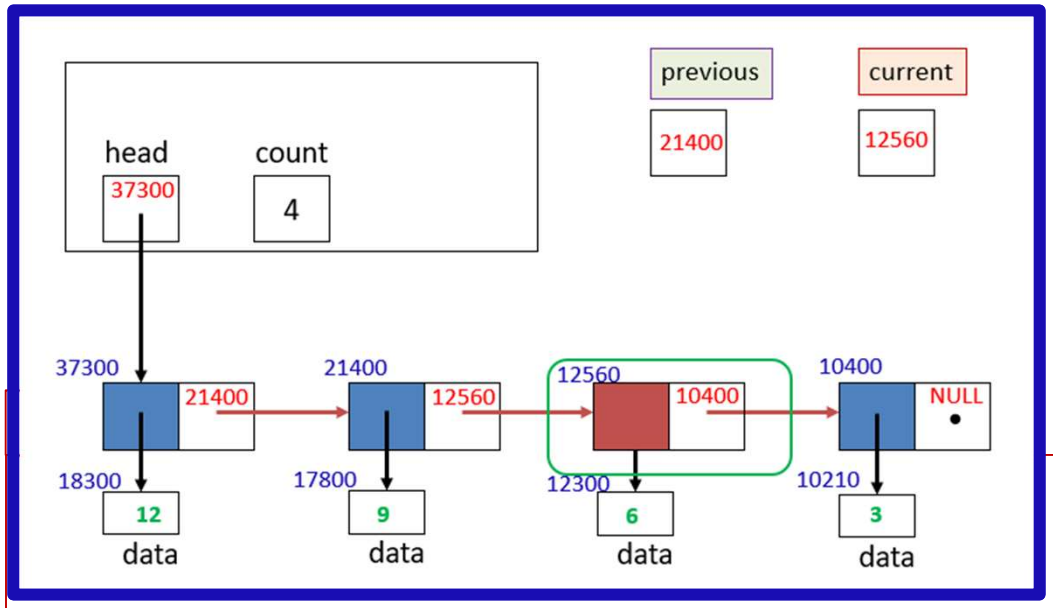

```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;   /*n is the first node*/
  else
    prev->next = cur->next;   /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	21400
prev	564 - 567	37300
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



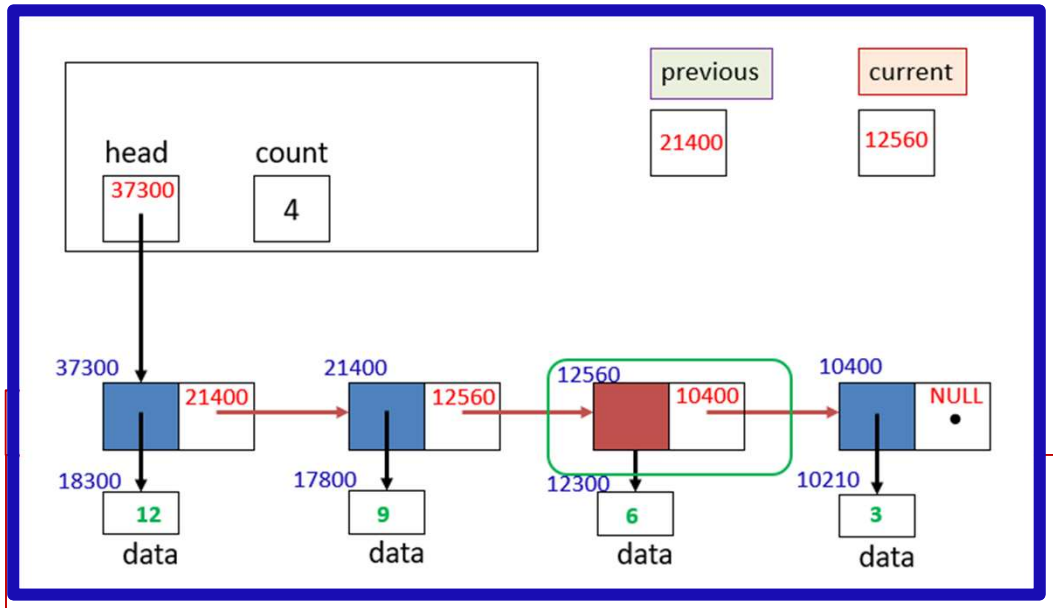

```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	21400
prev	564 - 567	37300
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



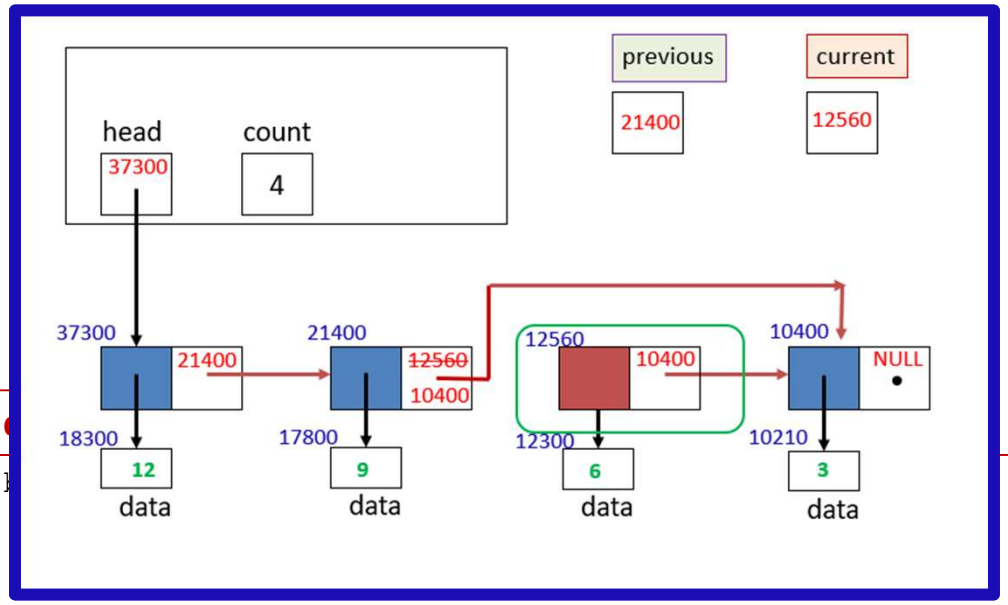

```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;   /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	12560
prev	564 - 567	21400
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



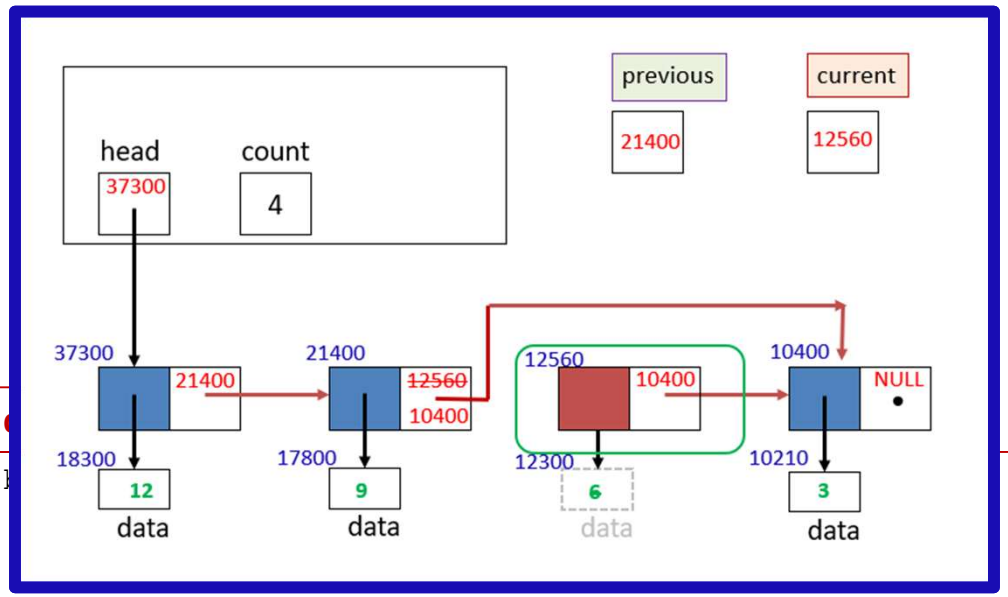

```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	12560
prev	564 - 567	21400
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	12560
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	



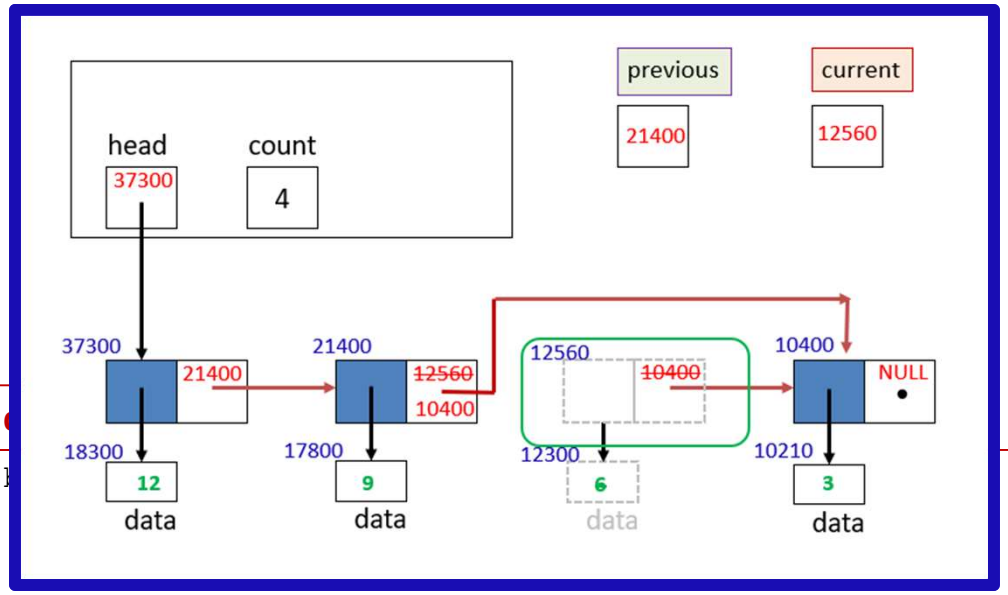

```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;   /*n is the first node*/
  else
    prev->next = cur->next;   /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	12560
prev	564 - 567	21400
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	10400
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	




```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	12560
prev	564 - 567	21400
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	10400
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	




```
  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list->head = cur->next;     /*n is the first node*/
  else
    prev->next = cur->next;     /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	12560
prev	564 - 567	21400
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
{ DM }	12300 - 12303	6
dataPtr	12560 - 12563	12300
next	12564 - 12567	10400
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	10400
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
    ...
    deleteValue(sList, 6);
    ...
}          // main
```

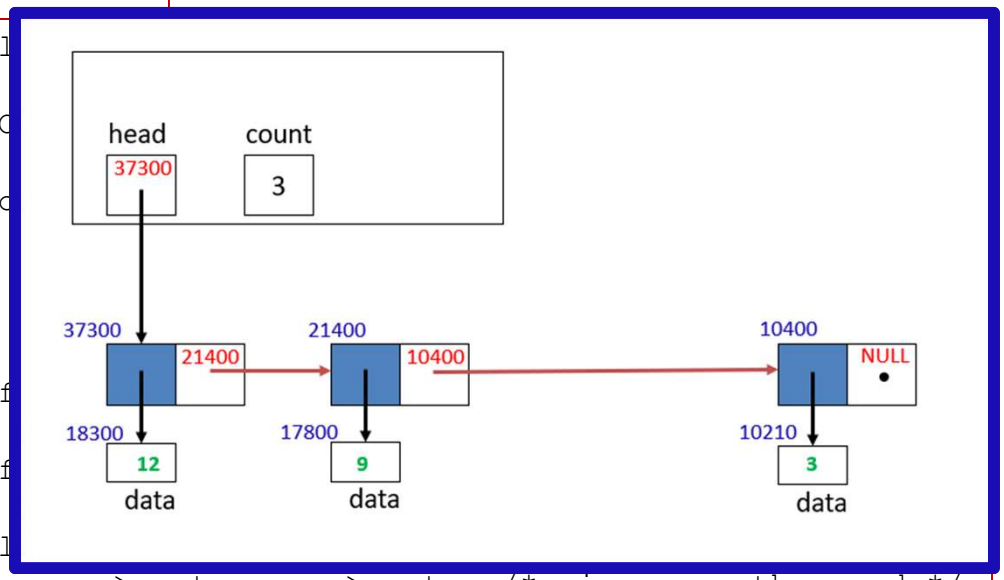
deleteValue.c

```
bool deleteValue(LIST* list, int value)
{
  NODE *cur, *prev;

  for (cur = list->head, prev = NULL;
       cur != NULL && *(int*)cur->dataPtr != value;
       prev = cur, cur = cur->next)
  ;  /* NOTICE the empty loop ! */

  if (cur == NULL)
    return list;              /* n was not found */
  if (prev == NULL)
    list->head = cur->next;   /*n is the first node*/
  else
    prev->next = cur->next;   /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
value	556 - 559	6
cur	560 - 563	12560
prev	564 - 567	21400
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	10400
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

```
    ...
  deleteValue(sList, 6);
    ...
}           // main
```

deleteValue.c




```
bool
{
  NC

  fo


  ;

  if

  if

  el
    prev->next = cur->next;  /*n is some other node*/

  free(cur->dataPtr);
  free(cur);

  return cur != NULL;
}
```

Label	Address	Value
newDataP	326 - 329	12300
sList	400 - 403	10100
	...	
	...	
	...	
	...	
head	10100 - 10103	37300
count	10104 - 10107	4
	...	
	...	
{ DM }	10210 - 10213	3
dataPtr	10400 - 10403	10210
next	10404 - 10407	NULL
	...	
	...	
	...	
{ DM }	17800 - 17803	9
dataPtr	21400 - 21403	17800
next	21404- 21407	10400
{ DM }	18300 - 18303	12
dataPtr	37300 - 37303	18300
next	37304- 37307	21400
	...	

Dynamic Linked Lists in C

END OF PART 6

Doubly-Linked Lists

- A common variation on linked lists is to have two pointers to other nodes within each node: one to the *next* node on the list, and one to the *previous* node
- Doubly-linked lists make some operations, such as deleting a tail node, more efficient
- Doubly-linked lists can have *iterators* for efficient forward and backward traversals
  - iterator can now have *operator++* and *operator- -*

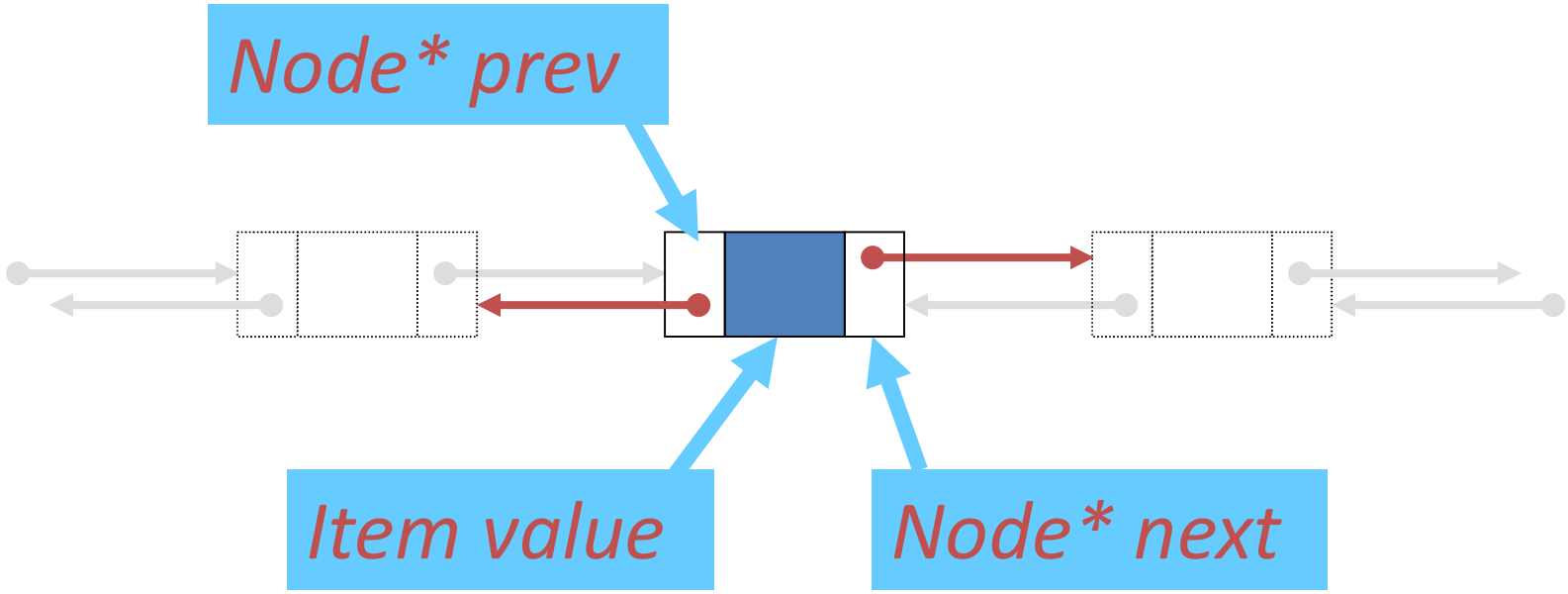
Doubly-Linked Lists

- Other operations, such as adding an item to an ordered linked list, are easier to program with doubly-linked lists

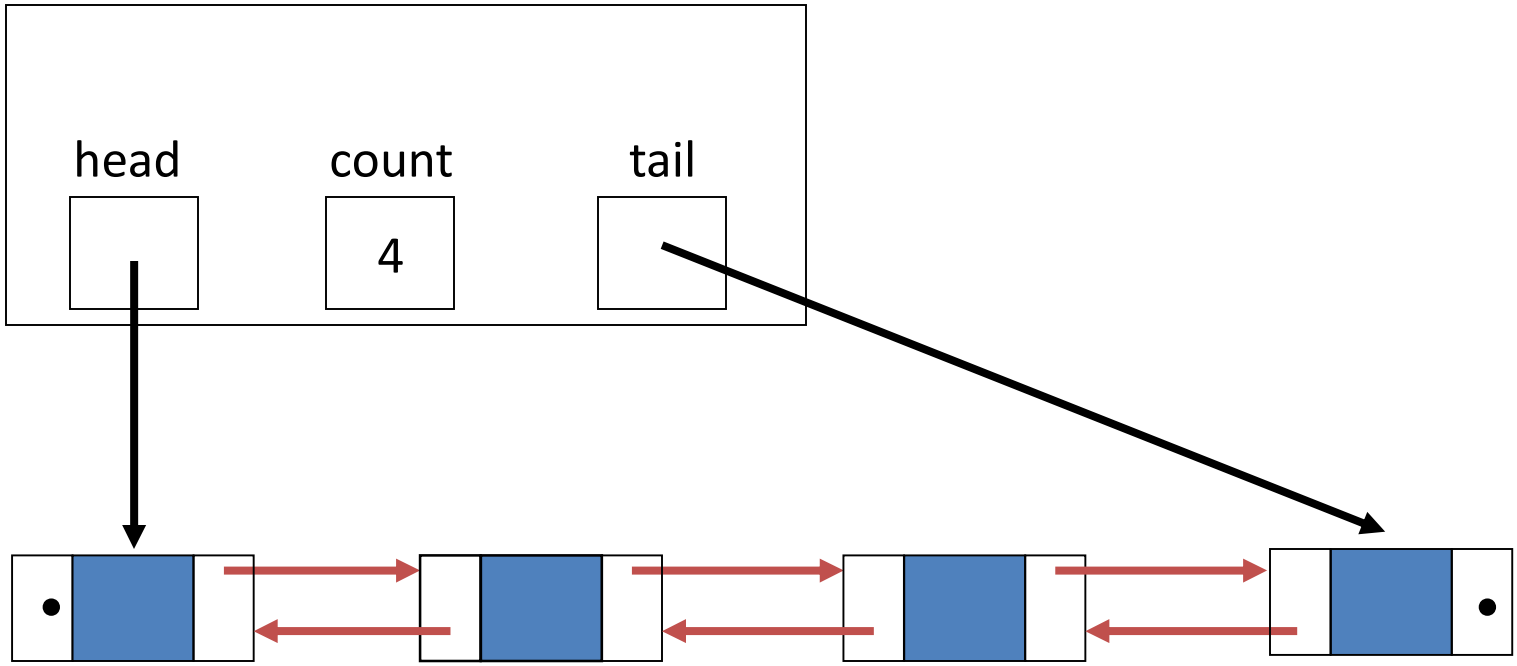
- *Tradeoffs*:
  - Each node requires
    4 (32 bit) additional bytes
  - or-
    8 (64 bit) additional bytes

Nodes in Doubly Linked Lists

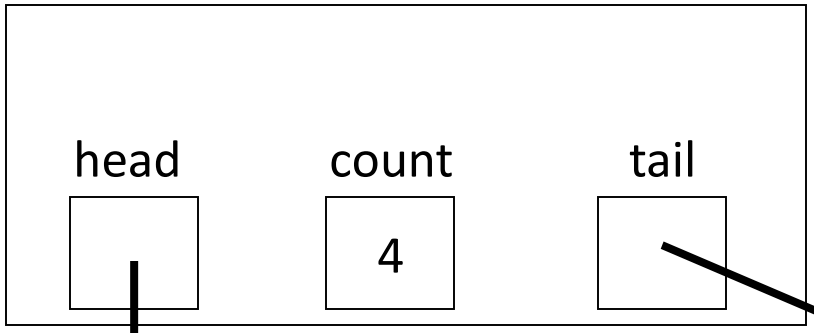
- Each *Node* object in a doubly linked list will contain three member variables:



Doubly-Linked List



To Remove Last Item From a Doubly-Linked List

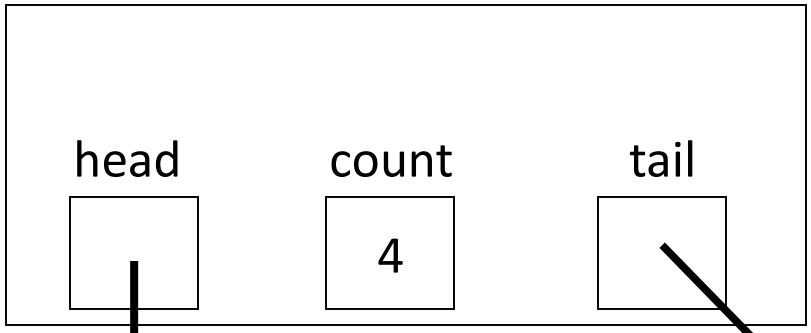


Note: We no longer need to traverse the list.

Save a copy of the last data object so that it can be returned later

(Copy of data item from last node)

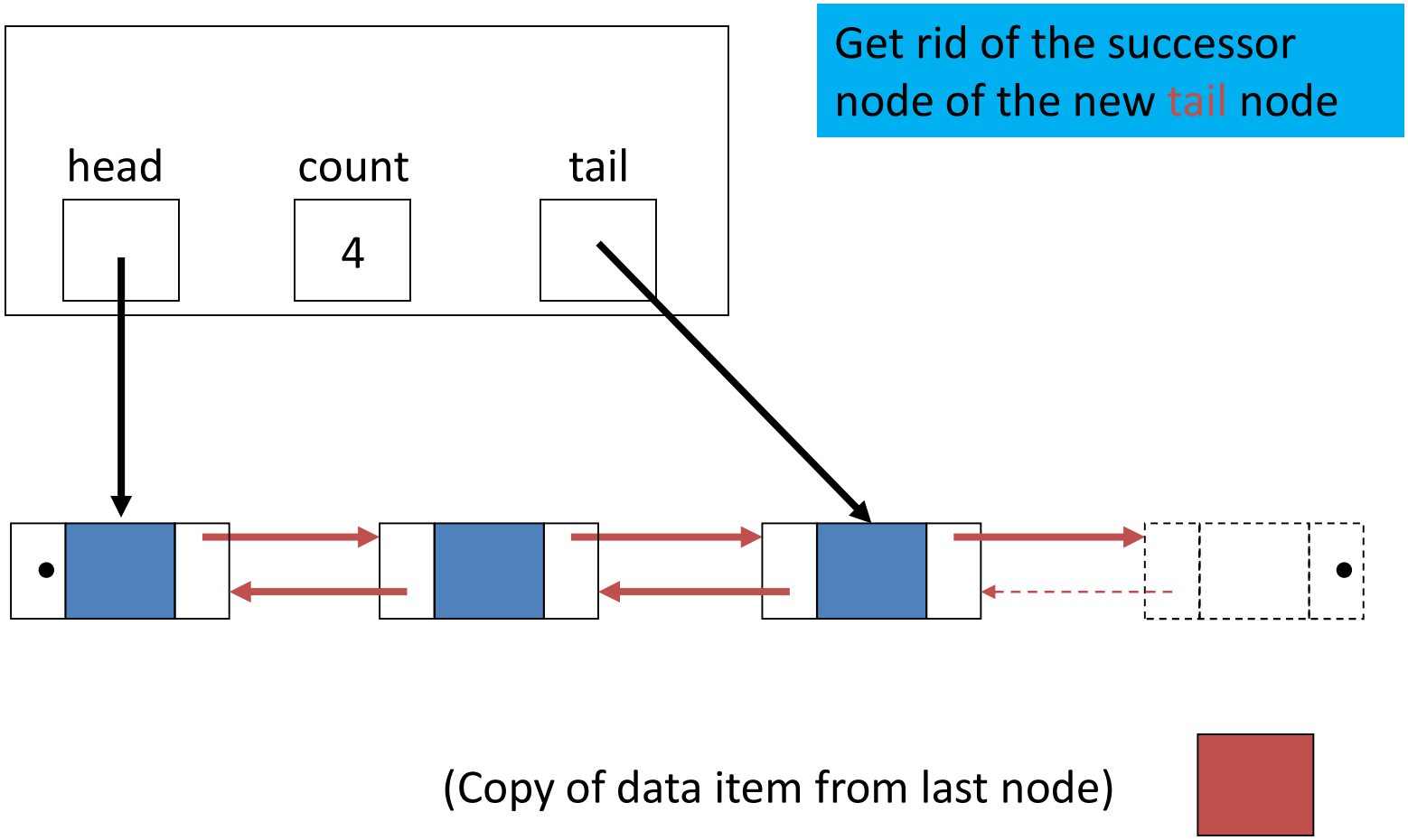
To Remove Last Item From a Doubly-Linked List



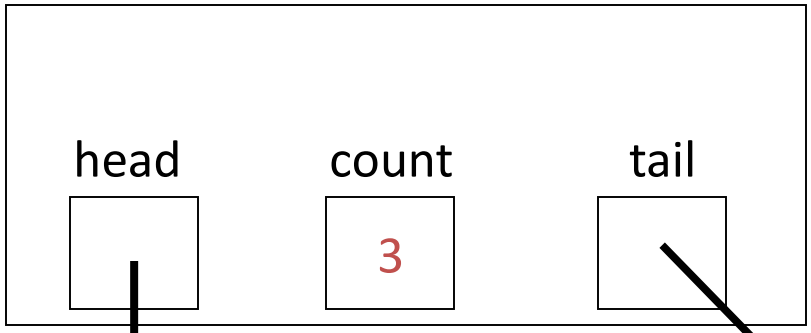
Reset tail so that it points at the node prior to the original tail node

(Copy of data item from last node)

To Remove Last Item From a Doubly-Linked List

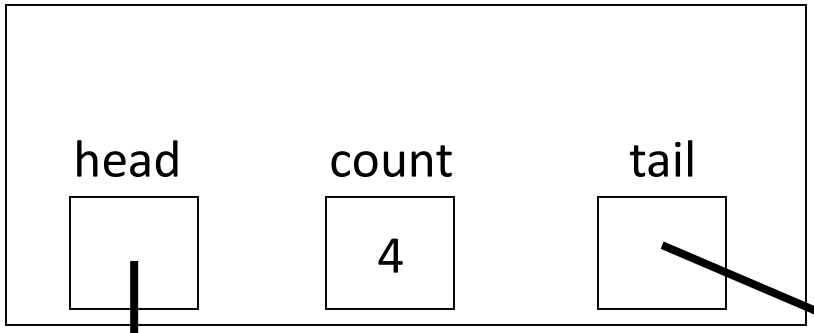


To Remove Last Item From a Doubly-Linked List



Set the successor of the new tail node to *NULL*, decrement the size of the list, and return the deleted data object

To Remove Middle Item From a Doubly-Linked List

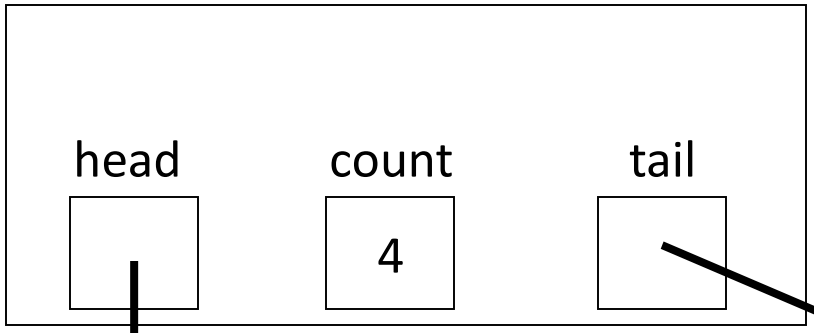


Note: We no longer need to traverse the list.

Save a copy of the last data object so that it can be returned later

(Copy of data item from node)

To Remove Middle Item From a Doubly-Linked List



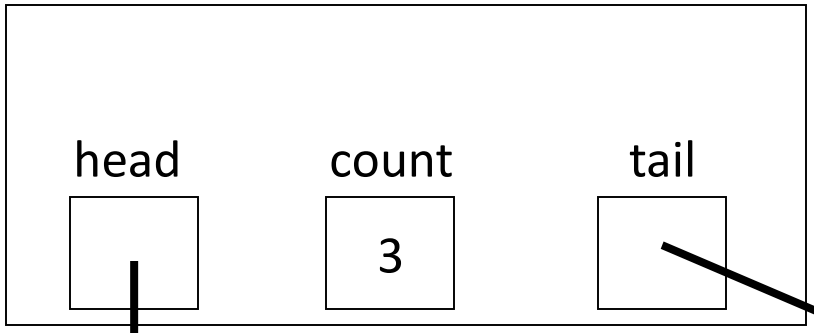
Note: We no longer need to traverse the list.

Save a copy of the last data object so that it can be returned later


```
cur->prev->next = cur->next ;
cur->next->prev = cur->prev ;

free(cur->dataPtr);
free(cur);
```

(Copy of data item from node)

To Remove Middle Item From a Doubly-Linked List



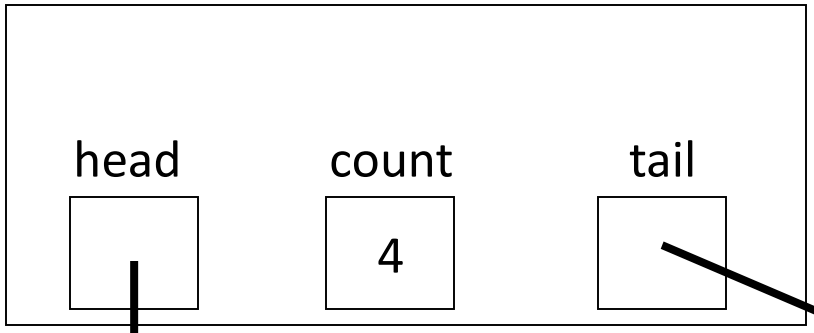
Note: We no longer need to traverse the list.

Save a copy of the last data object so that it can be returned later

```
cur->prev->next = cur->next ;
cur->next->prev = cur->prev ;

free(cur->dataPtr);
free(cur);
```

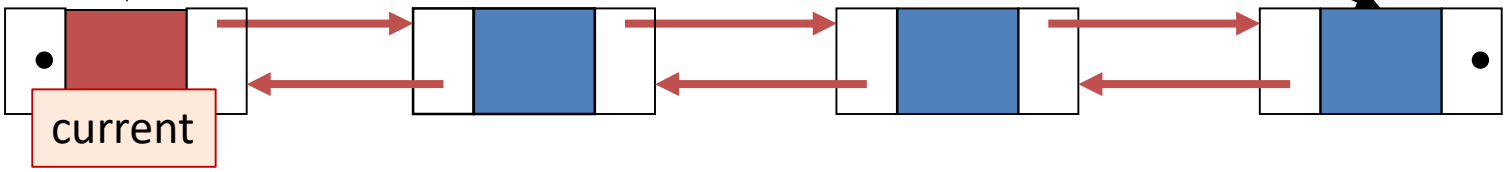
(Copy of data item from node)

To Remove First Item From a Doubly-Linked List



Note: We no longer need to traverse the list.

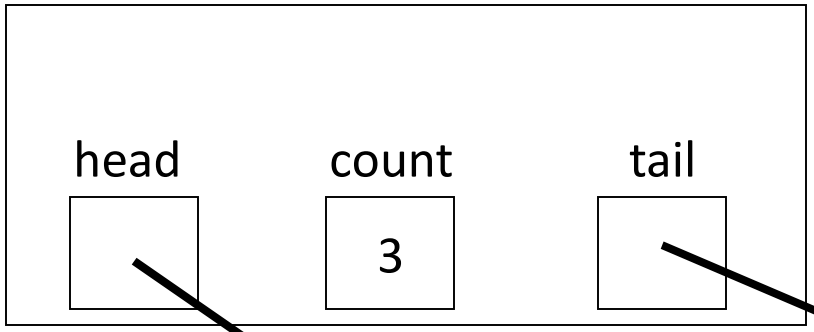
Save a copy of the last data object so that it can be returned later




```
head = cur->next ;
cur->next->prev = NULL;

free(cur->dataPtr);
free(cur);
```

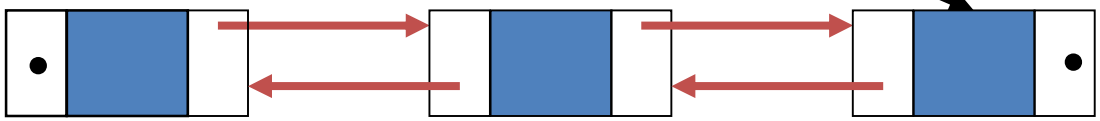
(Copy of data item from node)

To Remove First Item From a Doubly-Linked List



Note: We no longer need to traverse the list.

Save a copy of the last data object so that it can be returned later



```
head = cur->next ;
cur->next->prev = NULL;

free(cur->dataPtr);
free(cur);
```

(Copy of data item from node)

Dynamic Linked Lists in C

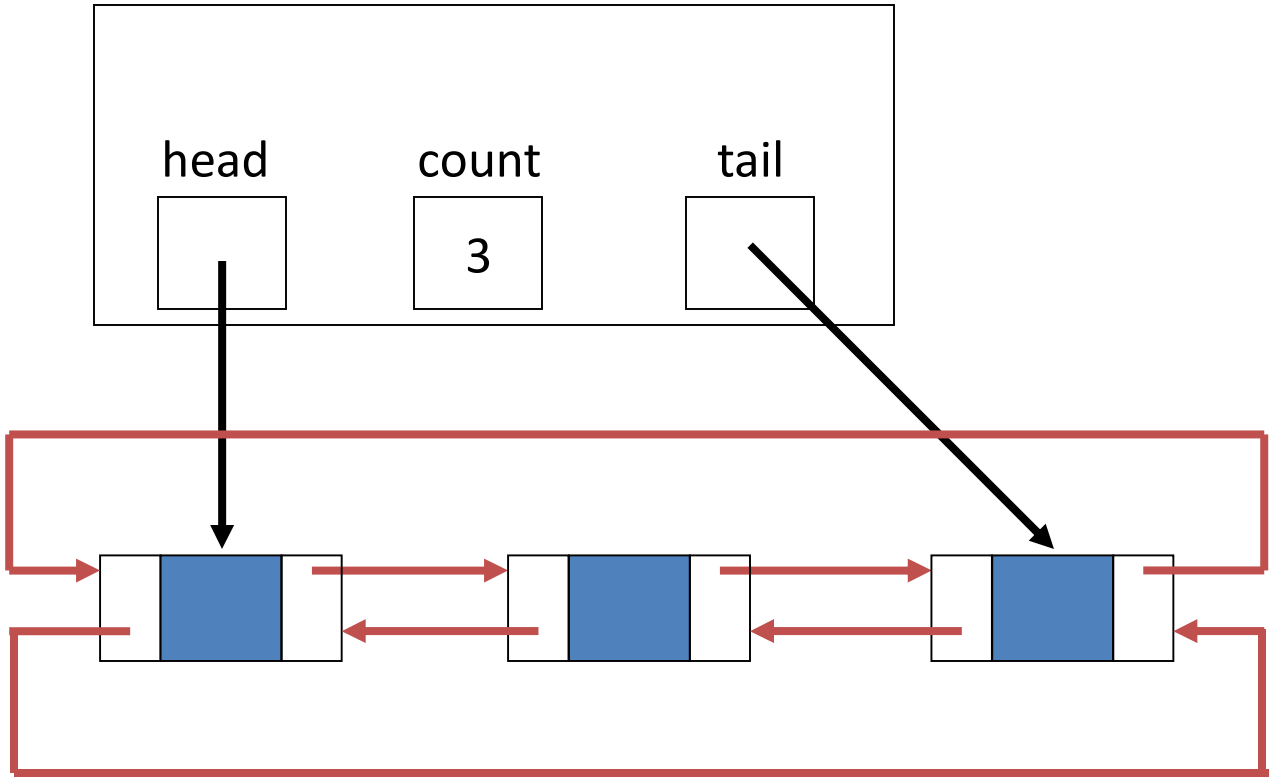
END OF PART 7

Circular Linked Lists

Circular Linked Lists

- Circular linked lists avoid the use of null references in their nodes
- Can be useful for certain algorithms
- Can also make programming simpler: fewer special cases to consider
- Successor of tail node is the head node; in doubly-linked list, predecessor of head node is the tail node

Circular Doubly-Linked List



Dynamic Linked Lists in C

END OF PART 8

END OF Dynamic Linked List