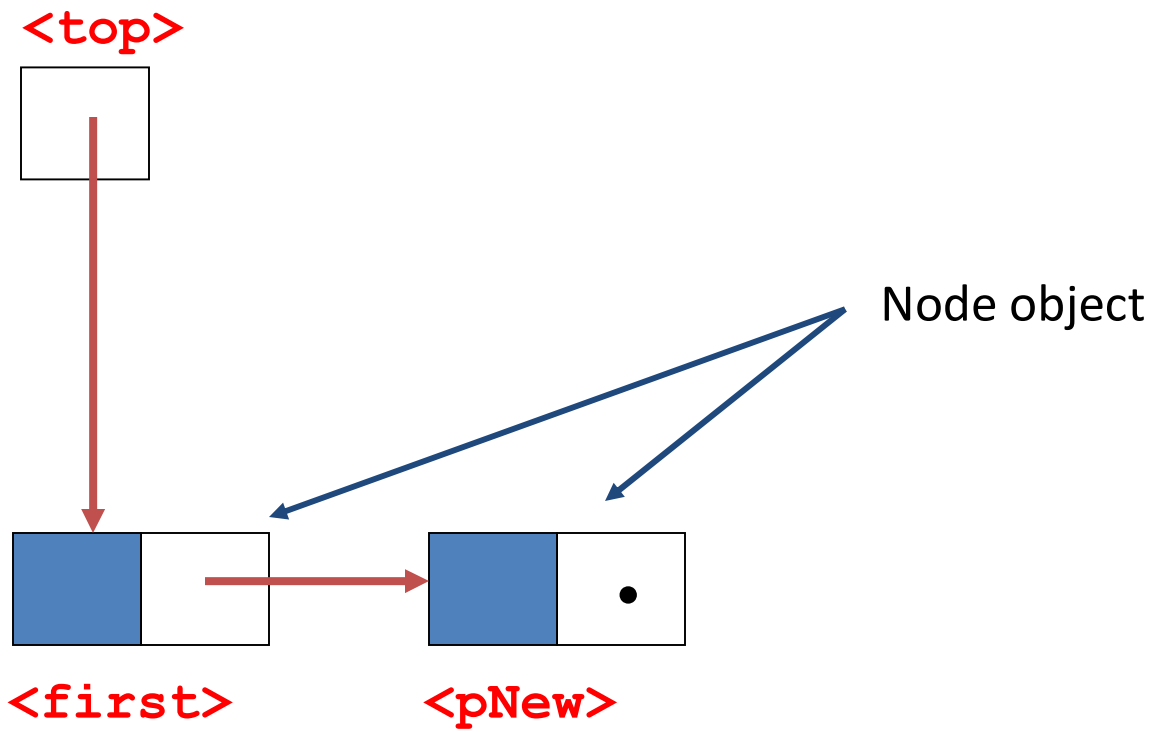
CS 2211
Systems Programming

Part Eleven:

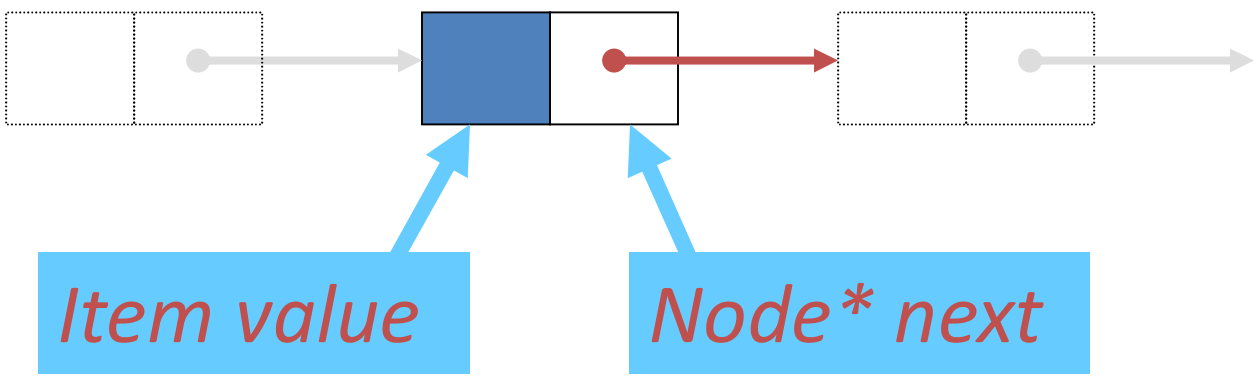
Linked Lists

Linked List

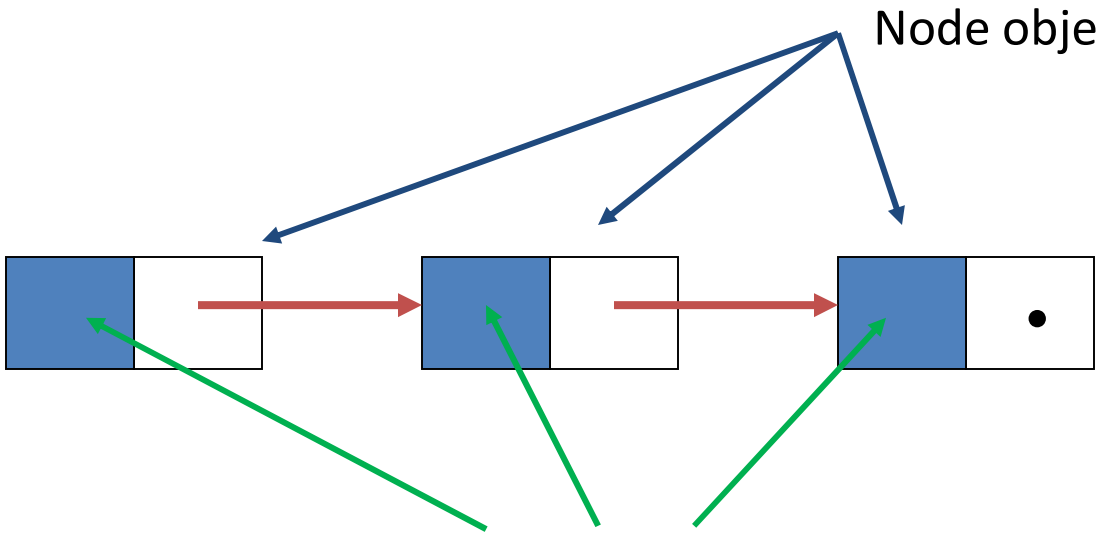


Nodes in Singly Linked Lists

- 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:



Linked List

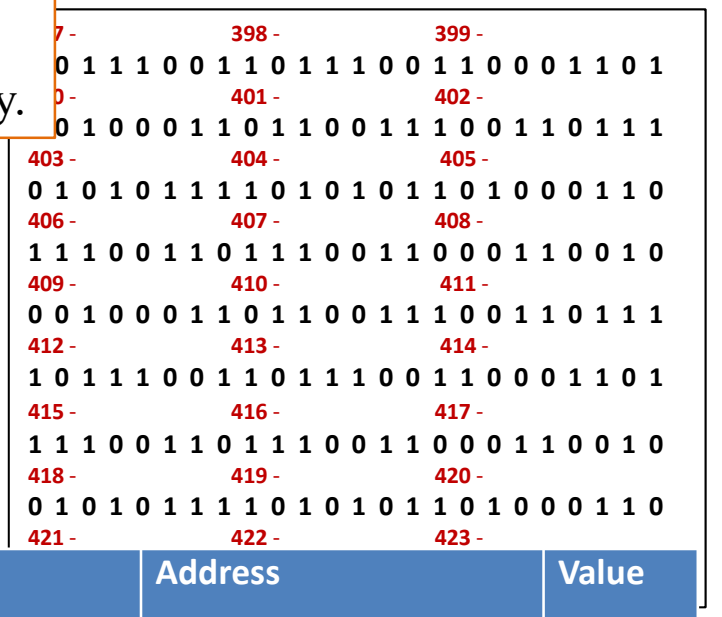


Data objects can be simple data (int, double, etc) or complex data (arrays, structures or unions) or pointers to data outside each individual nodes

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int   wP;
    int   fP;

     } fract;
```

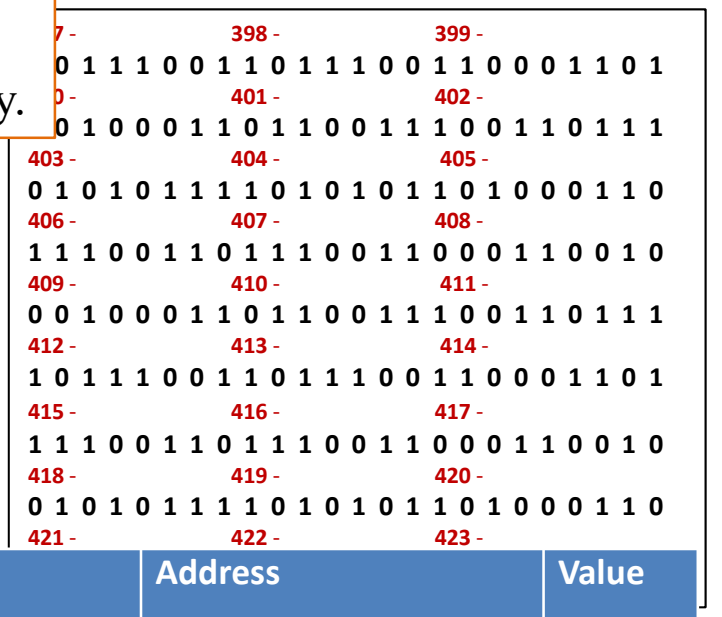


Address Label	Label	Address	Value

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;
```

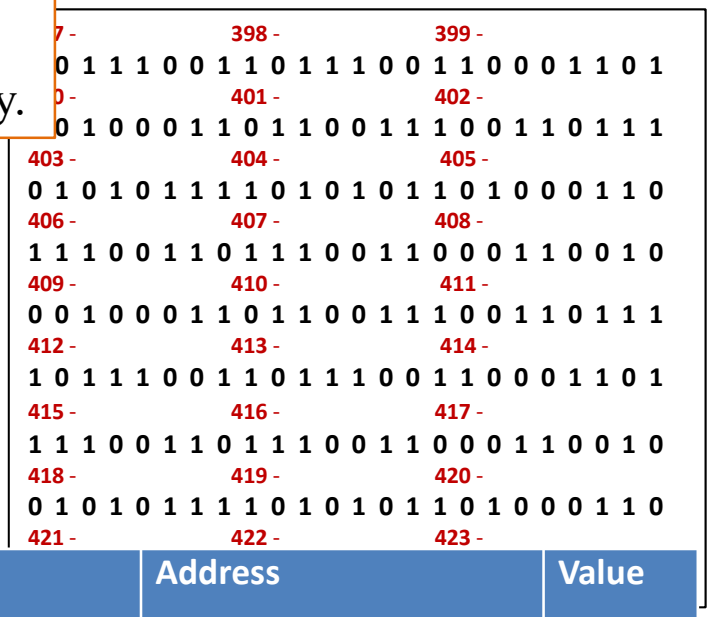


Address Label	Label	Address	Value

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first *pNew;
```

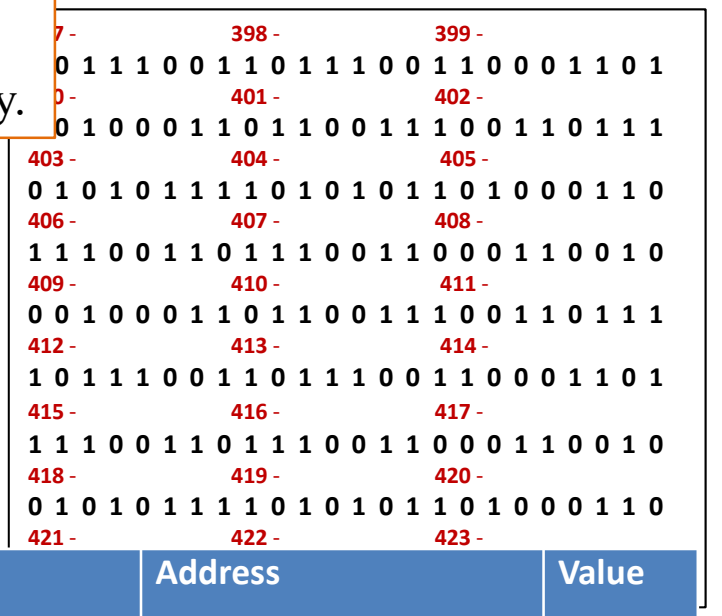


Address Label	Label	Address	Value
	first.wP	400 - 403	
	first.fP	404 - 407	
	first.link	408 - 411	
	pNew	412 - 415	

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first, *pNew;

first.wp = 7;
first.fP = 3;
```

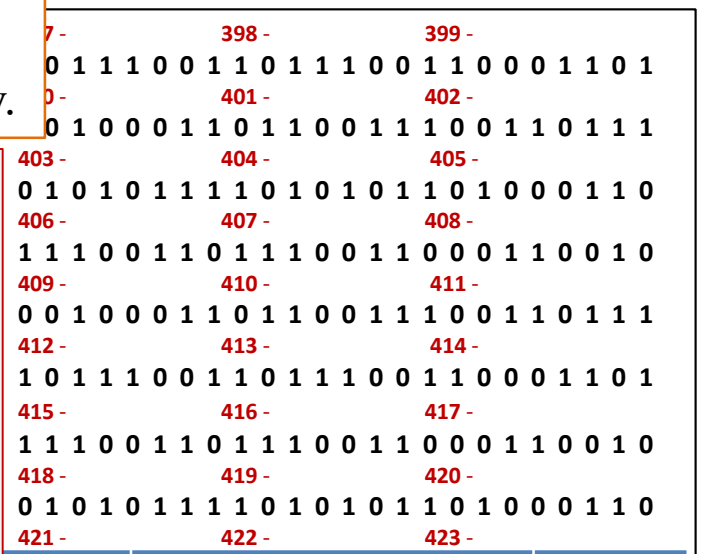


Address Label	Label	Address	Value
	first.wP	400 - 403	7
	first.fP	404 - 407	3
	first.link	408 - 411	
	pNew	412 - 415	

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first, *pNew;

first.wp = 7;
first.fP = 3;

pNew = (fract*)malloc(sizeof (fract));
```

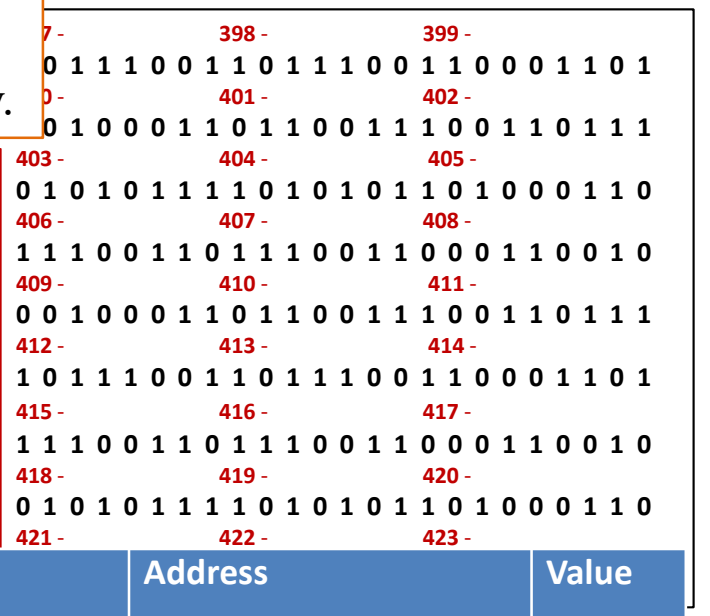


		Address	Value
		400 - 403	7
		404 - 407	3
		408 - 411	
	pNew	412 - 415	10100
	{DM}	10100 - 10111	

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first, *pNew;

first.wp = 7;
first.fP = 3;

pNew = (fract*)mal

pNew->wP =  56;
pNew->fP =  92;
```

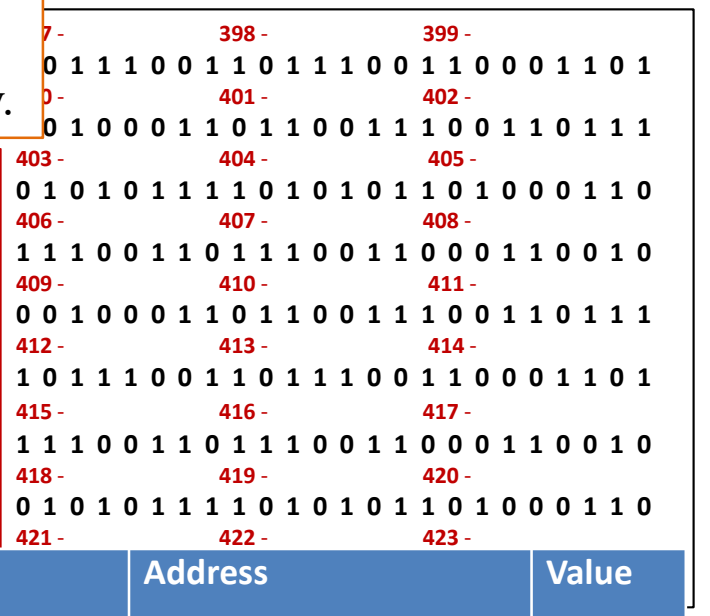


Address Label	Label	Address	Value
	first.wP	400 - 403	7
	first.fP	404 - 407	3
	first.link	408 - 411	
	pNew	412 - 415	10100
	pNew->wP	**10100 - 10103**	56
	pNew->fP	**10104 - 10107**	92
	{DM}	**10108 - 10111**	

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

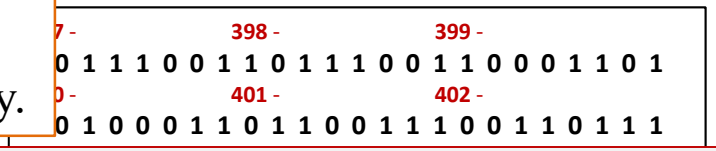
```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first, *pNew;

first.wp = 7;
first.fP = 3;

pNew = (fract*)mal

pNew->wP =  56;
pNew->fP =  92;

first.link =
     pNew;
```



Address Label	Label	Address	Value
	first.wP	400 - 403	7
	first.fP	404 - 407	3
	first.link	408 - 411	10100
	pNew	412 - 415	10100
	pNew->wP	10100 - 10103	56
	pNew->fP	10104 - 10107	92
	{DM}	10108 - 10111	

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

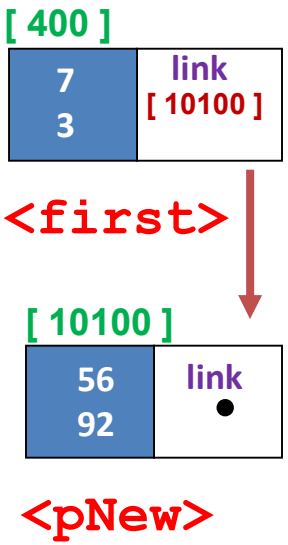



```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first, *pNew;

first.wp = 7;
first.fP = 3;

pNew = (fract*)malloc(sizeof (fract));

pNew->wP =  56;
pNew->fP =  92;

first.link = pNew;
printf( "\nwP: %d, fP: %d\n" , first.link->wP , first.link->fP ) ;
```

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS
- a pointer variable can also be used

```
...
pNew = (fract*)malloc(sizeof (fract));

pNew->wP =  56;
pNew->fP =  92;

first.link = pNew;
printf( "\nwP: %d, fP: %d\n" , first.link->wP , first.link->fP ) ;
```



Address Label	Label	Address	Value
	first.wP	400 - 403	7
	first.fP	404 - 407	3
	first.link	408 - 411	10100
	pNew	412 - 415	10100
	pNew->wP	10100 - 10103	56
	pNew->fP	10104 - 10107	92
	{DM}	10108 - 10111	

MEMORY MAPPING - STRUCTURE

better use of pointer variables in a link

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
     } fract;

fract first, *pTop, *pNew;

first.wp = 7;
first.fP = 3;

pTop = &first;

pNew = (fract*)malloc(sizeof (fract));

pNew->wP =  56;
pNew->fP =  92;

pTop->link = pNew;
printf( "\nwP: %d, fP: %d\n", pTop->link->wP , pTop->link->fP ) ;
```

Linked Lists in C

END OF PART 1