Intermediate Programming
Day 18

Outline

- Linked lists
- Review questions

Linked lists

We've seen some linked-list operations

- Create a node
- Add a node after a node
- Get the length of the list
- Print out the contents

We need some more:

- Add to the front of the list
- Remove an element from the list
- Deallocate memory associated to the list
- Make a copy of the list

```
charList.h
#ifndef charList_included
#define charList_included
typedef struct _Node
{
     struct _Node *next;
     char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

Linked lists

- Insertion
  - Create the linked-list element
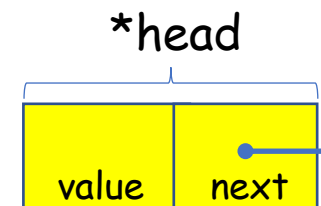 - Update the pointers

charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

charList.h

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

*head

value	next

Linked lists

- Insertion
  - Create the linked-list element
  - Update the pointers

charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

charList.h

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

n   *head

| value | NULL next | | value | next |

Linked lists

- Insertion
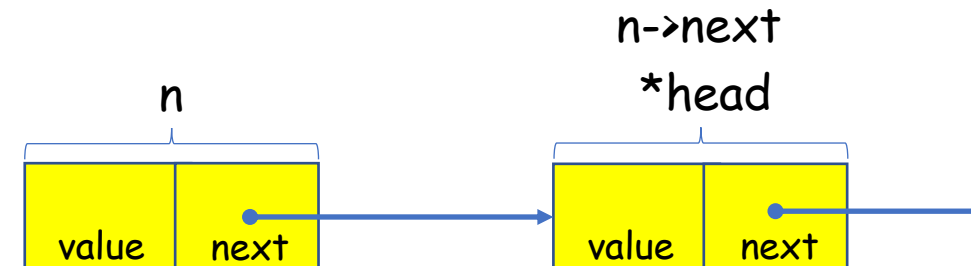 - Create the linked-list element
  - Update the pointers

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

n->next
*head

n

| value | next |

| value | next |

Linked lists

- Insertion
  - Create the linked-list element
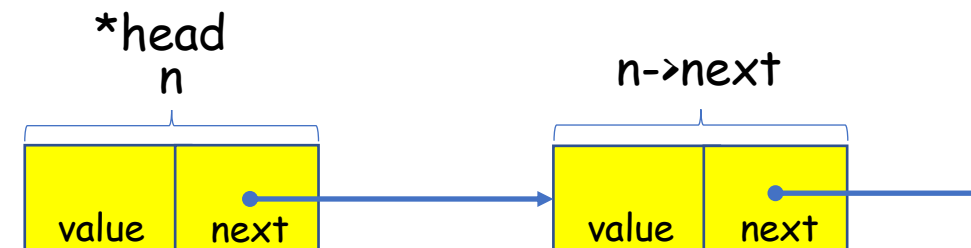 - Update the pointers

charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

charList.h

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
int length( const Node *head );
void print( const Node *head );
…
#endif // charList_included
```

Linked lists

- Insertion
  - Create the linked-list element
  - Update the pointers

```c
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```
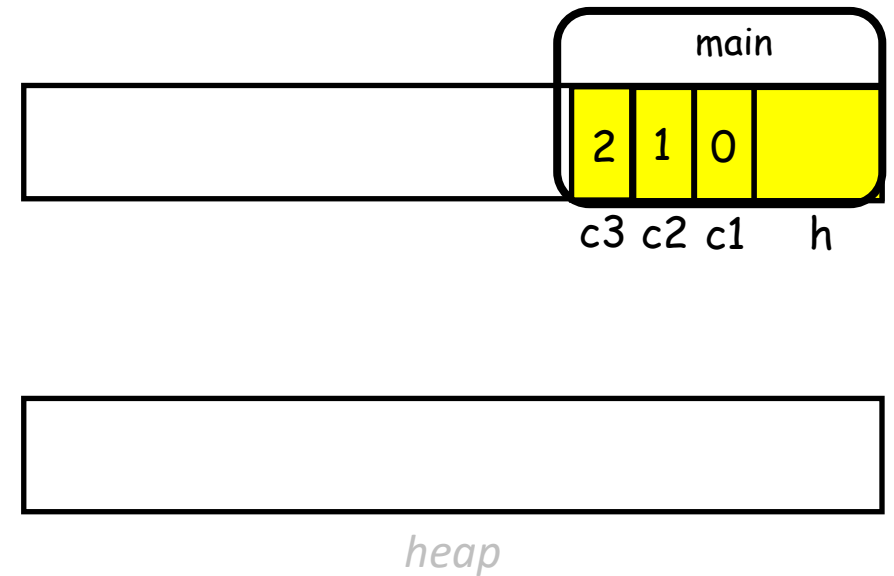
charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

main

| 2 | 1 | 0 | |

c3 c2 c1   h

heap

Linked lists

- Insertion
  - Create the linked-list element
  - Update the pointers

charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

main.c

```c
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```

Linked lists

- Insertion
  - Create the linked-list element
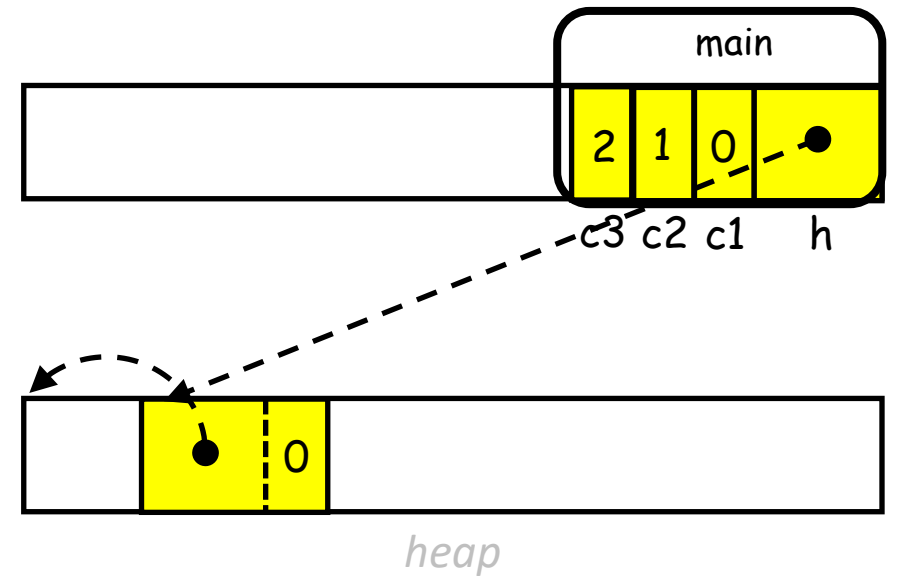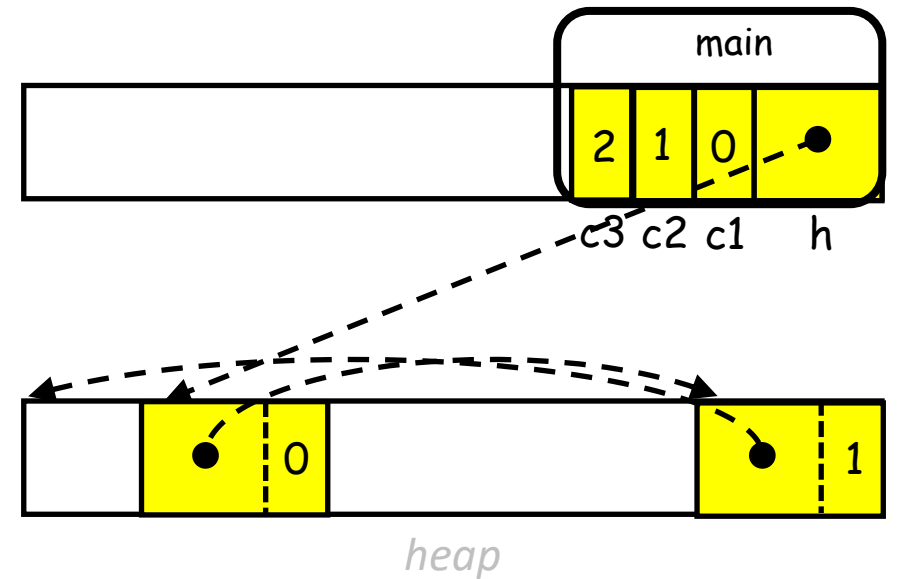 - Update the pointers

```c
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```

charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```


10

Linked lists

- Insertion
  - Create the linked-list element
  - Update the pointers

```
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

```
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```

Linked lists

- Insertion
  - Create the linked-list element
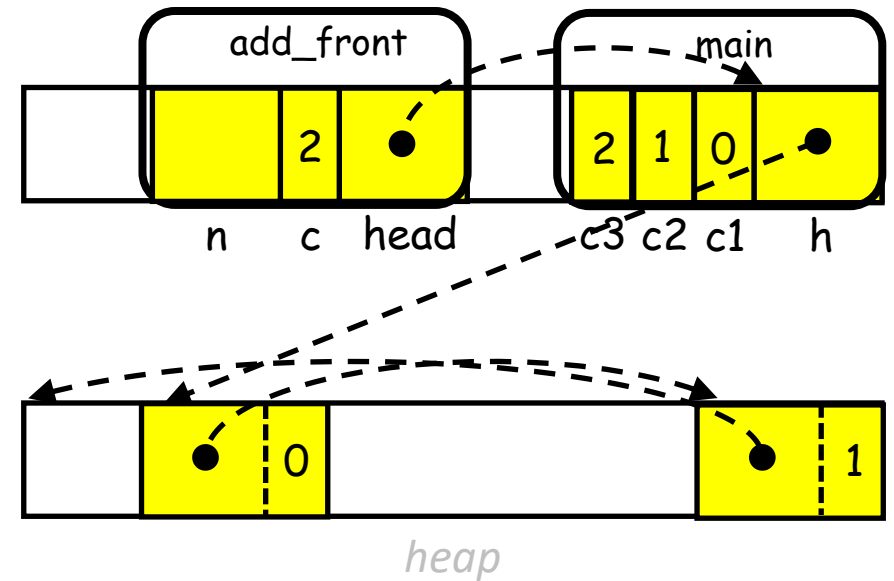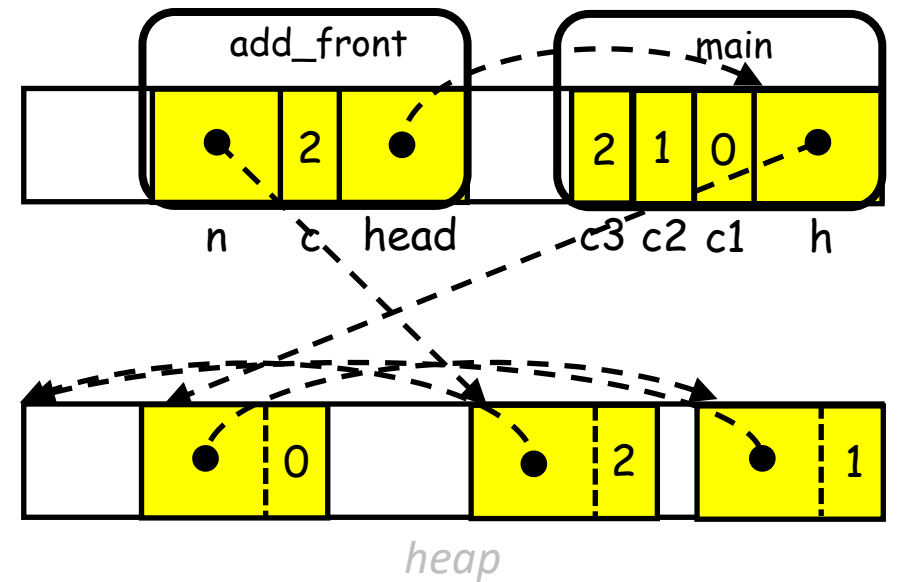 - Update the pointers

```c
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```

charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```


12

Linked lists

- Insertion
  - Create the linked-list element
  - Update the pointers

charList.c

```
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

main.c

```
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```

Linked lists

- Insertion
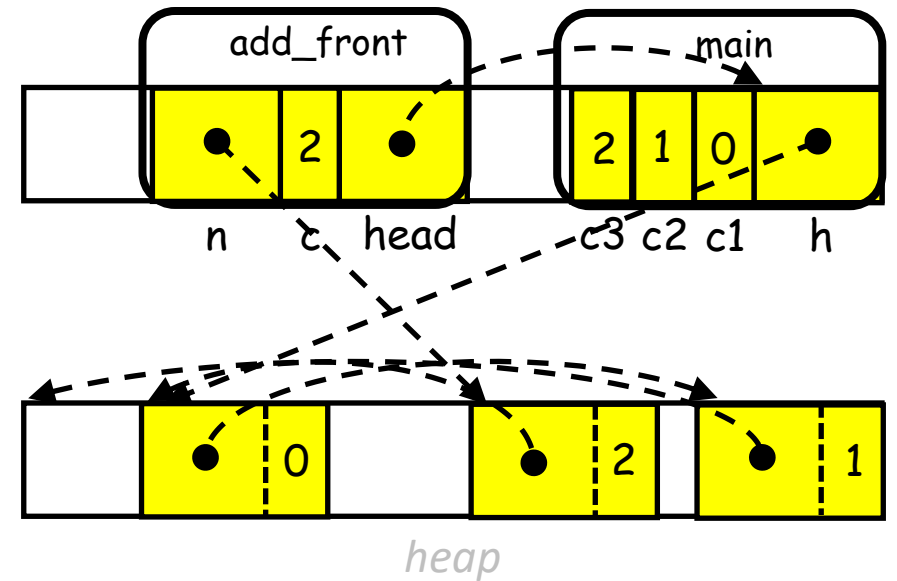 - Create the linked-list element
  - Update the pointers

```
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

```
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```

Linked lists

- Insertion
  - Create the linked-list element
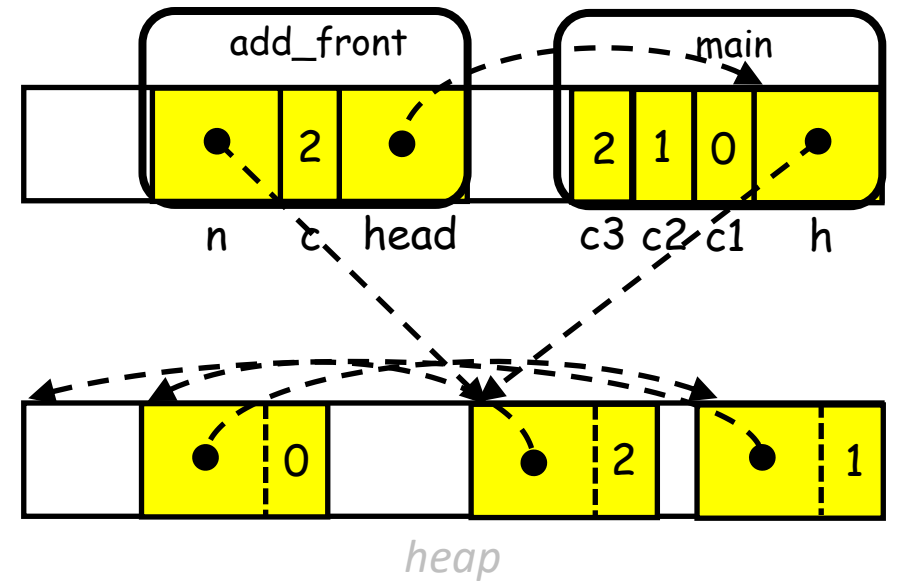 - Update the pointers

```c
#include "charList.h"

void main( void )
{
    char c1=0 , c2=1 , c3=2;
    Node *h = create_node( c1 );
    add_after( h , c2 );
    add_front( &h , c3 );
    return 0;
}
```
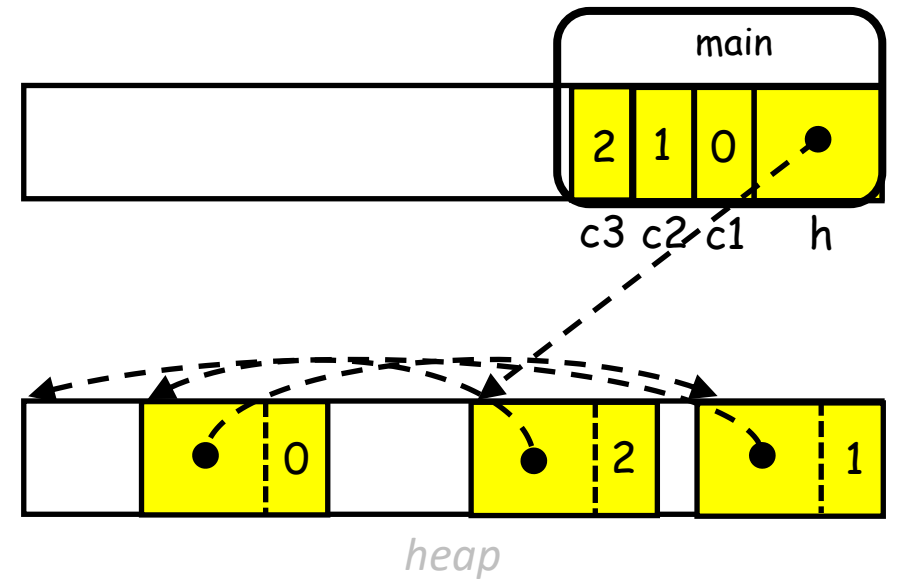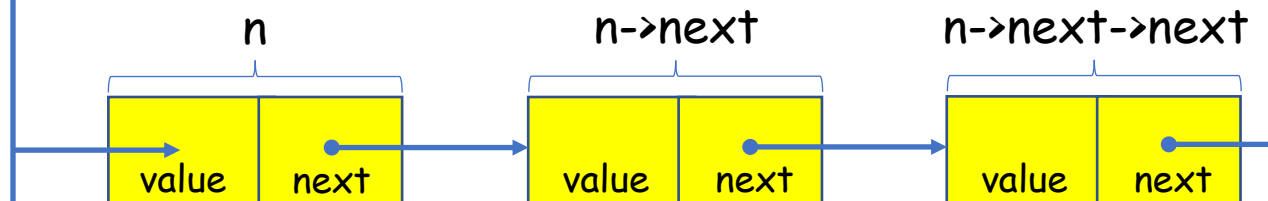
charList.c

```c
#include "charList.h"
…
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```


15

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.c

```
#include "charList.h"
…
void remove_after( Node *n )
{
    Node *nNext = n->next;
    if( !nNext ) return;
    n->next = n->next->next;
    free( nNext );
}
```

charList.h

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

n   n->next   n->next->next

| value | next | | value | next | | value | next |

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

```
#include "charList.h"
…
void remove_after( Node *n )
{
    Node *nNext = n->next;
    if( !nNext ) return;
    n->next = n->next->next;
    free( nNext );
}
```

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

n->next
nNext

n   n->next->next

| value | next | | value | next | | value | next |

Linked lists

- Deletion
  - Update the pointers
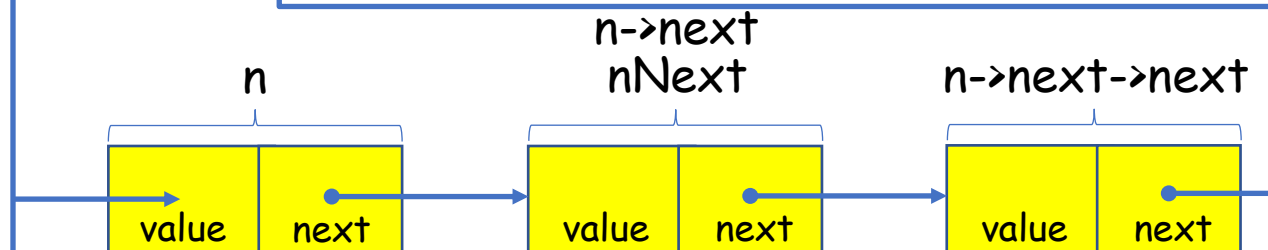 - Delete the linked-list element

charList.c

```
#include "charList.h"
…
void remove_after( Node *n )
{
    Node *nNext = n->next;
    if( !nNext ) return;
    n->next = n->next->next;
    free( nNext );
}
```

charList.h

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

n   nNext   n->next

value	next
value	next
value	next

Linked lists

- Deletion
  - Update the pointers
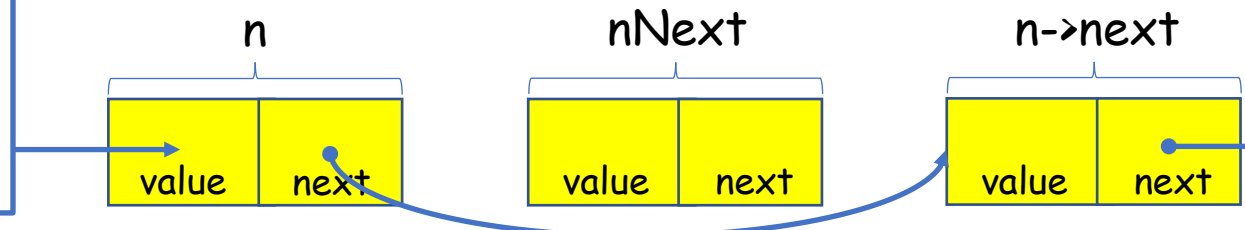 - Delete the linked-list element

charList.c

```c
#include "charList.h"
…
void remove_after( Node *n )
{
    Node *nNext = n->next;
    if( !nNext ) return;
    n->next = n->next->next;
    free( nNext );
}
```

charList.h

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

n

n->next

value	next

value	next

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.c

```
#include "charList.h"
…
void remove_front( Node **head )
{
    Node* n = (*head);
    if( !n ) return;
    *head = n->next;
    free( n );
}
```

charList.h

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
void remove_front( Node **n );
int length( const Node *head );
void print( const Node *head );
#endif // charList_included
```

*head

(*head)->next

| value | next |

| value | next |

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
void remove_front( Node **n );
int length( const Node *head );
void print( const Node *head );
#endif // charList_included
```

charList.c

```
#include "charList.h"
…
void remove_front( Node **head )
{
    Node* n = (*head);
    if( !n ) return;
    *head = n->next;
    free( n );
}
```

n
*head

n->next
(*head)->next

value   next   value   next

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.c

```
#include "charList.h"
…
void remove_front( Node **head )
{
    Node* n = (*head);
    if( !n ) return;
    *head = n->next;
    free( n );
}
```

charList.h

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
void remove_front( Node **n );
int length( const Node *head );
void print( const Node *head );
#endif // charList_included
```

n

n->next
*head

| value | next |

| value | next |

Linked lists

- Deletion
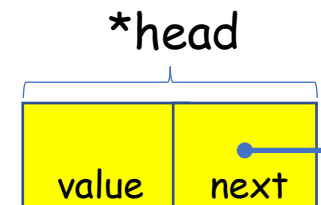 - Update the pointers
  - Delete the linked-list element

```c
#include "charList.h"
…
void remove_front( Node **head )
{
    Node* n = (*head);
    if( !n ) return;
    *head = n->next;
    free( n );
}
```

charList.h
```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
void remove_front( Node **n );
int length( const Node *head );
void print( const Node *head );
#endif // charList_included
```

*head

value	next

Linked lists

- Copying
  - Create a new node with the head's value
  - Make it's next a (recursive) deep copy of the remainder of the list

```
#include "charList.h"
…
Node *copy( const Node *head )
{
    if( !head ) return NULL;
    Node *_head = create_node( head->value );
    _head->next = copy( head->next );
    return _head;
}
```

```
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
void remove_front( Node **n );
int length( const Node *head );
void print( const Node *head );
Node *copy( const Node *head );
#endif // charList_included
```

Linked lists

Example (sorting `char`s)

- Read in `char`s from the `stdin` and insert them into a linked list, sorted from smallest to largest
  - Read the `char`s in
    - If the linked list is empty, create a head containing the `char`
    - Otherwise, if the `char` is smaller than everything in the linked list, add it at the head
    - Otherwise, add it after the largest element smaller than the `char`
  - Print out the (sorted) `char`s
  - Free up the memory

```
                        charList.h
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;

Node *create_node( char c );
int add_after( Node *n , char c );
int add_front( Node **h , char c );
void remove_after( Node *n );
void remove_front( Node **n );
int length( const Node *head );
void print( const Node *head );
Node *copy( const Node *head );
#endif // charList_included
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```

```
>> ./a.out
misha
ahims
>>
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
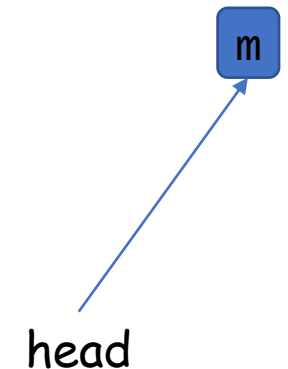
```
>> ./a.out
```

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
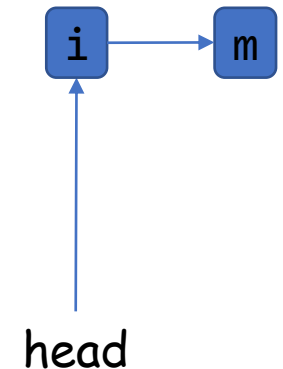
```
>> ./a.out
m
```

m

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
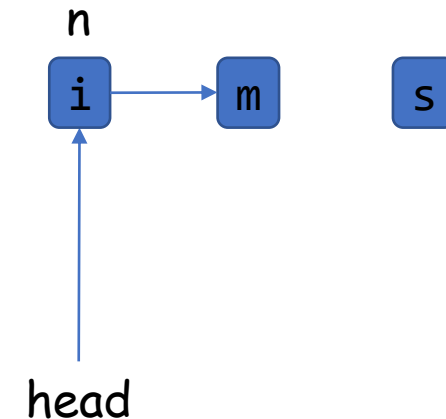
```
>> ./a.out
mi
```

i → m

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
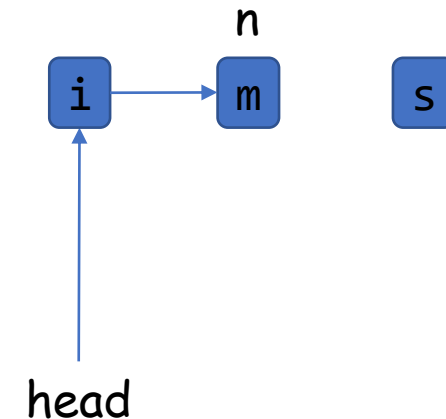
```
>> ./a.out
mis
```

n

i → m   s

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
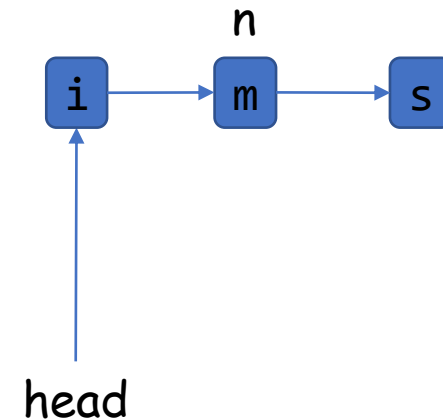
```
>> ./a.out
mis
```

n

i → m   s

head

31

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
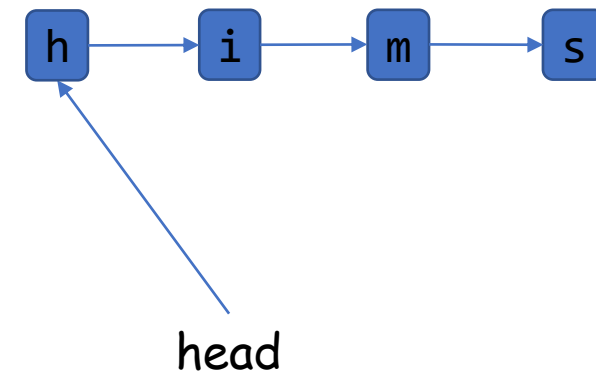
```
>> ./a.out
mis
```

n

i   m   s

head

32

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
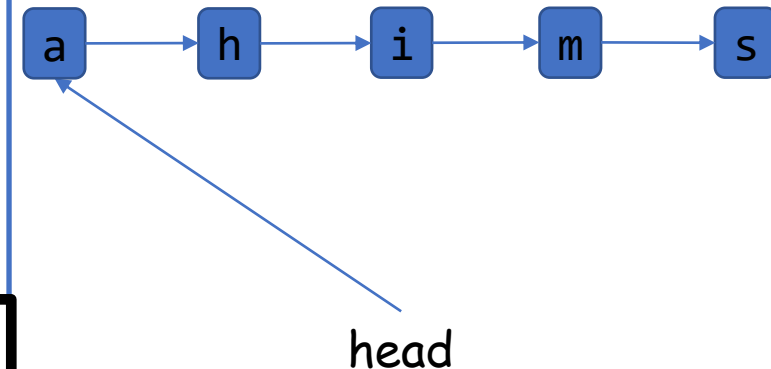
```
>> ./a.out
mish
```

h → i → m → s

head

33

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
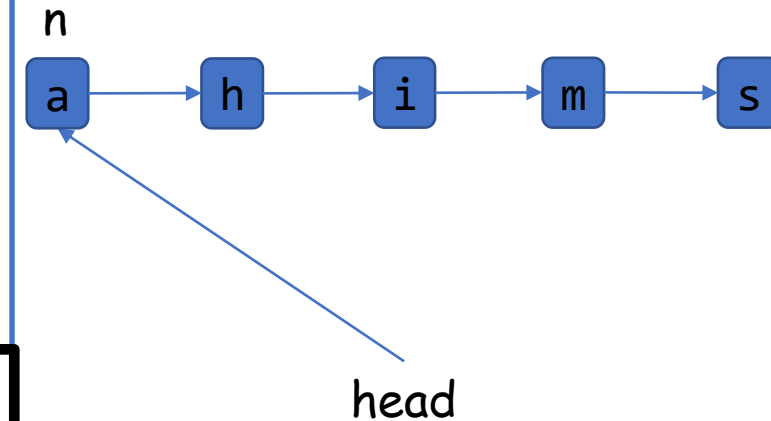
```
>> ./a.out
misha
```


a → h → i → m → s

head

34

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
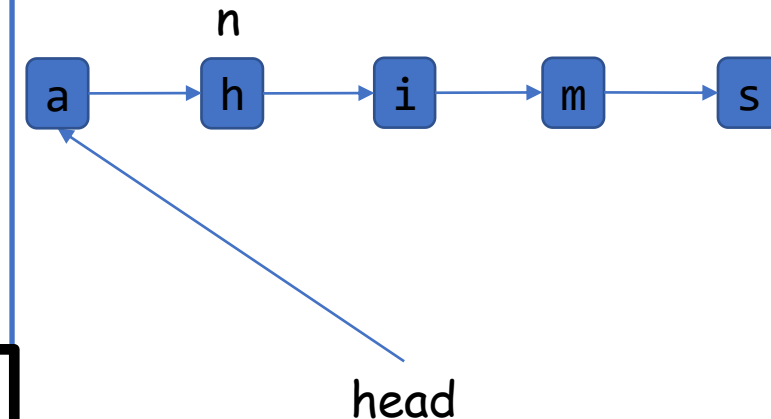
n

a → h → i → m → s

head

```
>> ./a.out
misha
a
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
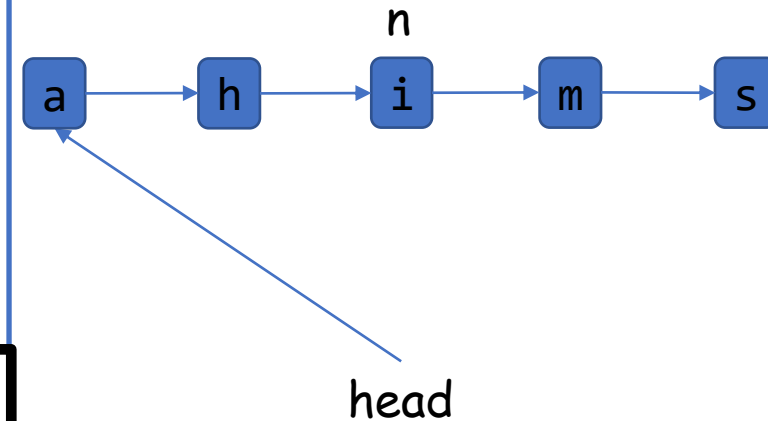
```
>> ./a.out
misha
ah
```

n

a → h → i → m → s

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{

    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
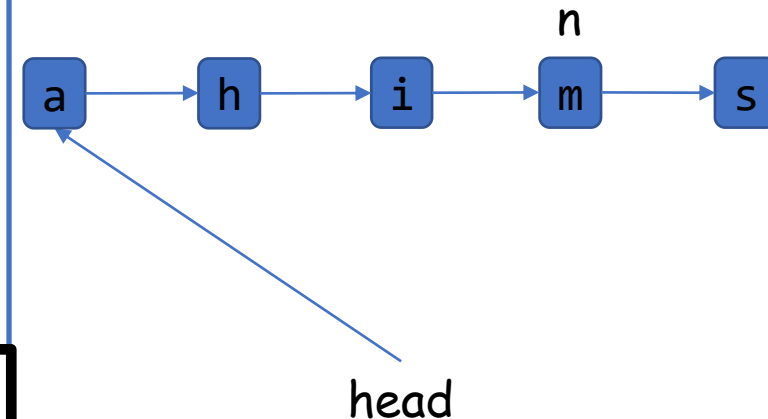
```
>> ./a.out
misha
ahi
```

n

a → h → i → m → s

head

37

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
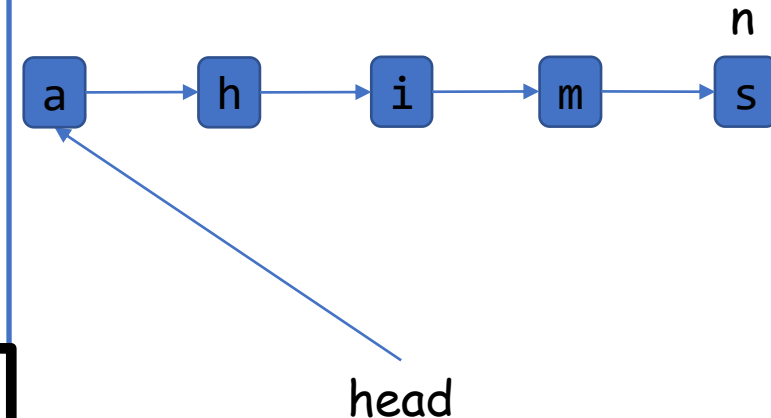
```
>> ./a.out
misha
ahim
```

n

a → h → i → m → s

head

38

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
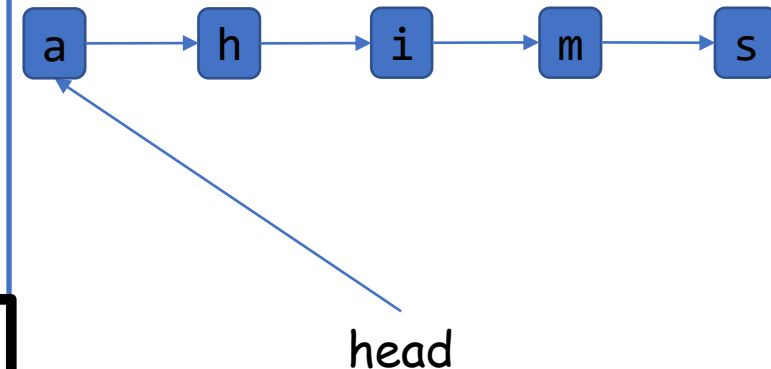


```
>> ./a.out
misha
ahims
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
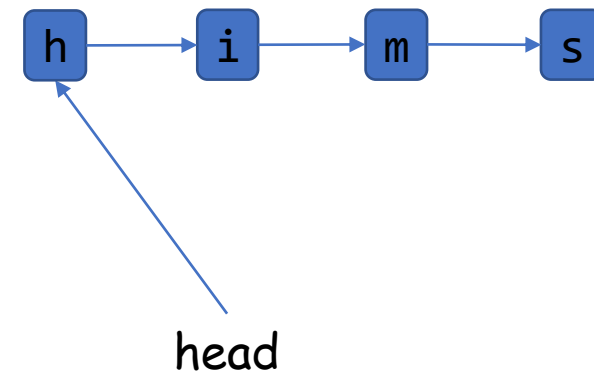

a → h → i → m → s

head

```
>> ./a.out
misha
ahims
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
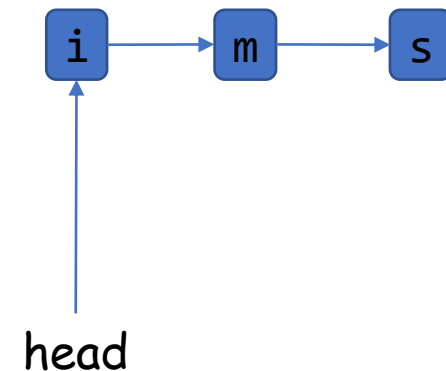


```
>> ./a.out
misha
ahims
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```


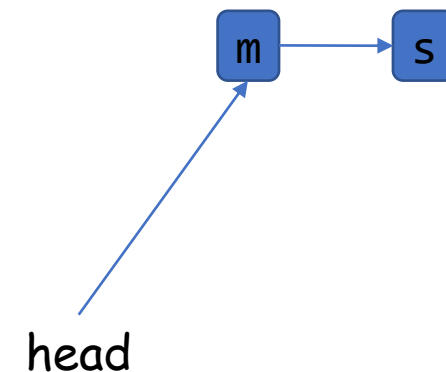
```
>> ./a.out
misha
ahims
```

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```

```
>> ./a.out
misha
ahims
```

m → s

head

43

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```
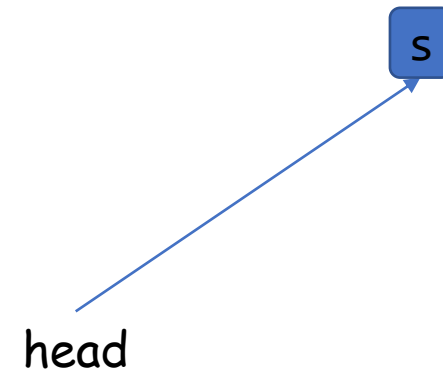
```
>> ./a.out
misha
ahims
```

s

head

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
int main( void )
{
    Node *head = NULL , *n;
    char c;
    while( fscanf( stdin , " %c" , &c )==1 )
    {
        if( !head ) head  = create_node( c );
        else if( c<head->value ) add_front( &head , c );
        else
        {
            for( n=head ; n->next!=NULL && c>=n->next->value ; n=n->next ) ;
            add_after( n , c );
        }
    }
    for( n=head ; n!=NULL ; n=n->next ) printf( "%c" , n->value );
    printf( "\n" );
    while( head ) remove_front( &head );
    return 0;
}
```

```c
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```

```
>> ./a.out
misha
ahims
>>
```

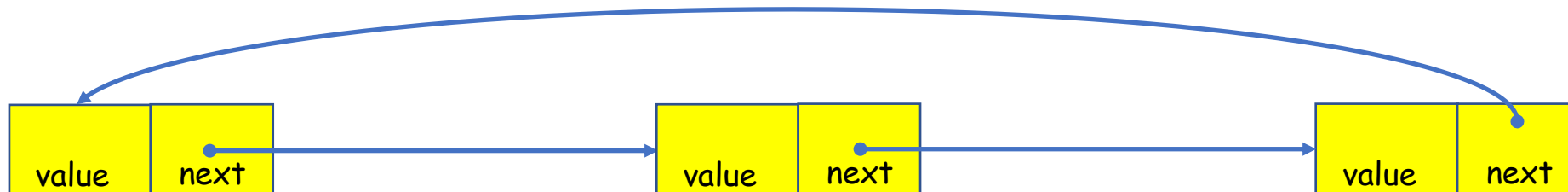head

45

Linked lists

- Variants
  - Circular lists
    - ✓ No need for a "head" node
    - ✗ Iterating is trickier

```
                              charList.h
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    char value;
} Node;
...
#endif // charList_included
```

Linked lists

- Variants
  - Doubly linked lists
    - ✓ Can traverse in either direction
    - ✗ More pointers to track for insertions and deletions
    - ✗ The linked list can be inconsistent

```
                                    charList.h
#ifndef charList_included
#define charList_included
typedef struct _Node
{
    struct _Node *next;
    struct _Node *prev;
    char value;
} Node;
...
#endif // charList_included
```

value	prev	next

value	prev	next

value	prev	next
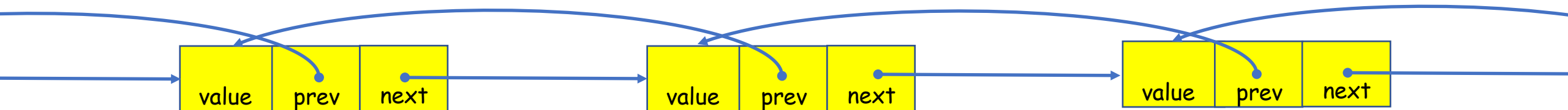
Outline

- Linked lists
- **Review questions**

Review questions

1.How do you implement add_front of a linked list?

```
int add_front( Node **head , char c )
{
    Node *n = create_node( c );
    if( !n ) return 1;
    n->next = *head;
    *head = n;
    return 0;
}
```

Review questions

2. How do you modify a linked list to a doubly linked list?

```
typedef struct _Node
{
    struct _Node *next , *prev;
    char value;
} Node;
```

Review questions

3. How do you make a copy of a linked list?

We need a "deep copy". We traverse the list and create new node from the old one. We need to pay attention to how to setup the next pointer for the new list. It should point to the newly created node.

Review questions

4. Why does add_after take a Node* as input, but add_front takes a Node**?

Because we need to change who the head is.

Review questions

5. What cases should be handled when implementing `remove_front`?

Check if the list is empty (the head is NULL).

Exercise 18

- Website -> Course Materials -> Exercise 18