Intermediate Programming

Day 17

Outline

- Exercise 6-1
- Linked lists
- Review questions

Exercise 6-1 (part 1)

Implement the `length` function.

```
unsigned int length( const Node *head )
{
    unsigned int len = 0;
    while( head ) len++ , head = head->next;
    return len;
}
```

Exercise 6-1 (part 2)

Implement the `add_after` function.

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

Exercise 6-1 (part 3)

Implement the **reverse_print** function.

```
void reverse_print( const Node *node )
{
    if( node->next ) reverse_print( node->next );
    printf( "%c " , node->data );
}
```

Outline

- Exercise 6-1
- 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

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 length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

Linked lists

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

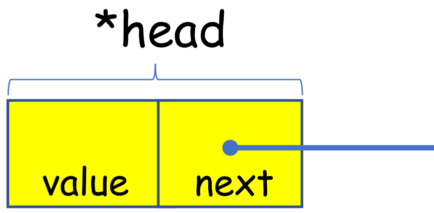
charList.c

```
#include "charList.h"
#include "assert.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

```
#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

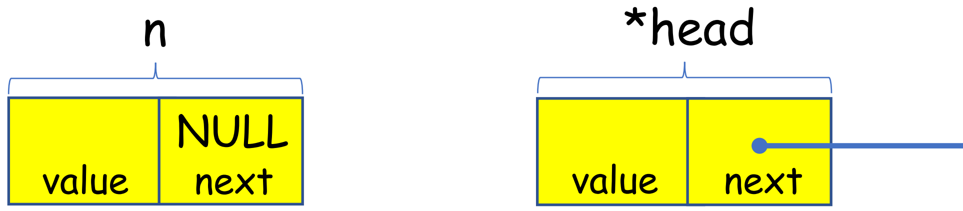
charList.c

```
#include "charList.h"
#include "assert.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

```
#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

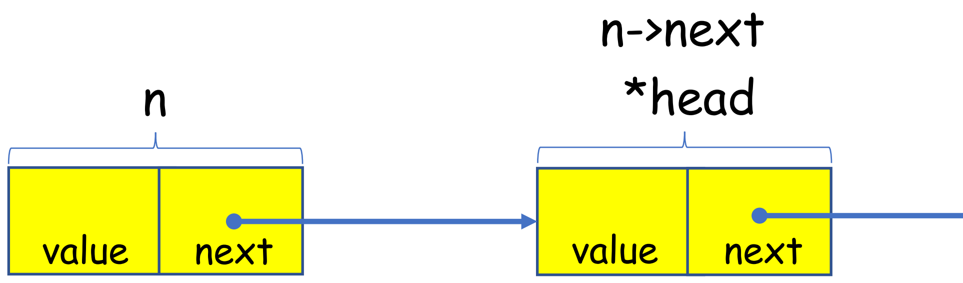
charList.c

```
#include "charList.h"
#include "assert.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

```
#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

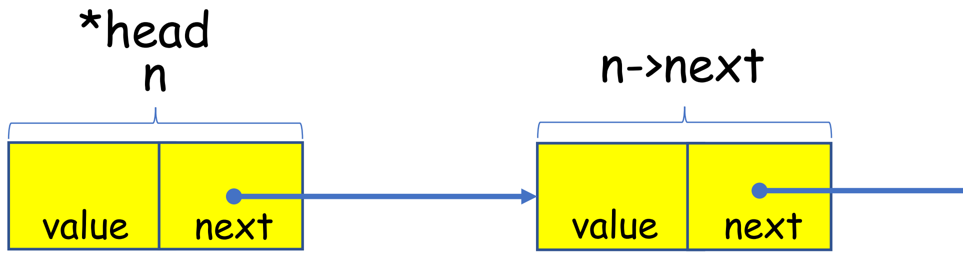
charList.c

```
#include "charList.h"
#include "assert.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

```
#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
```



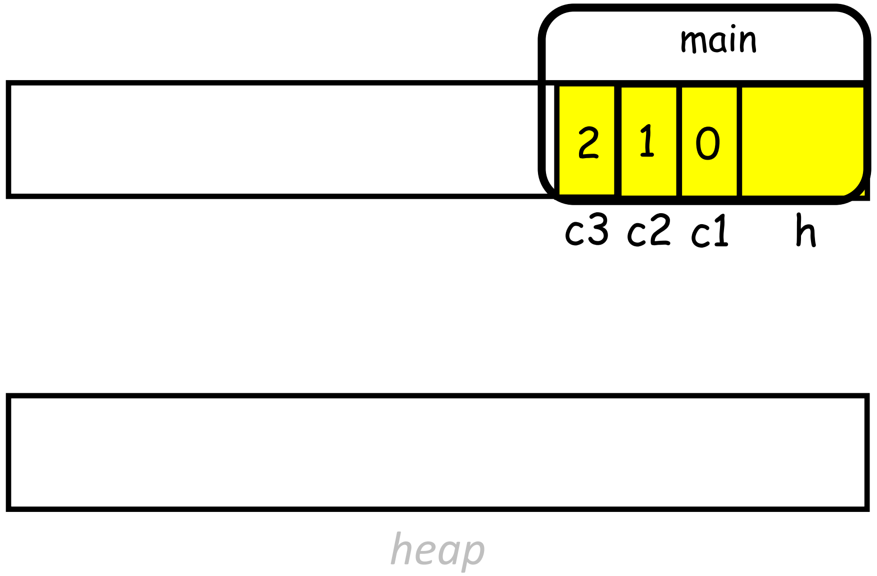
Linked lists

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

```
                    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;
}
```

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



Linked lists

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

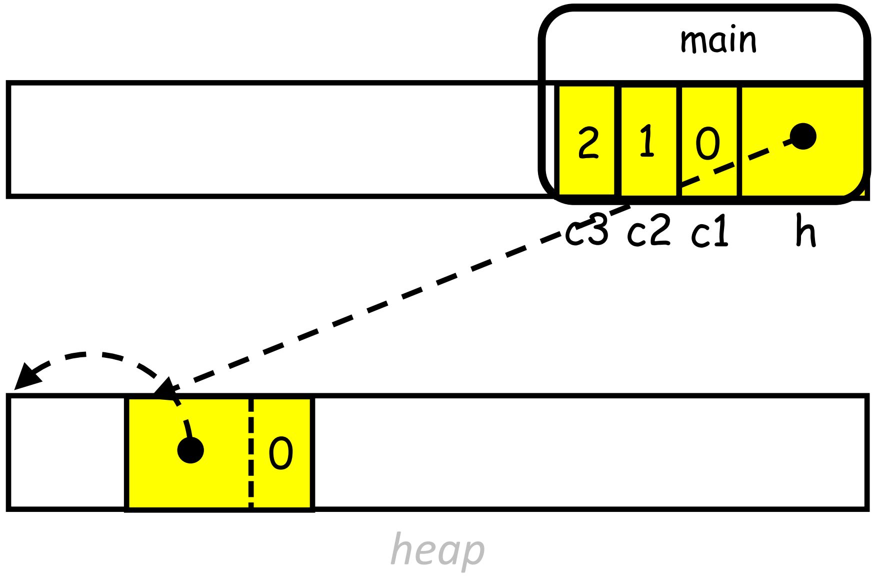
charList.c

```
#include "charList.h"
#include "assert.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

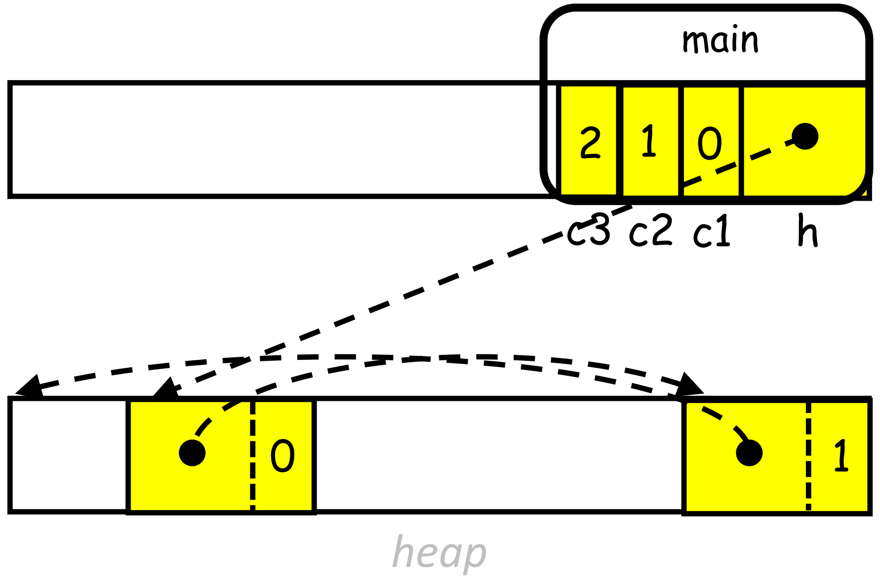
main.c

```
#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;
}
```

charList.c

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



Linked lists

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

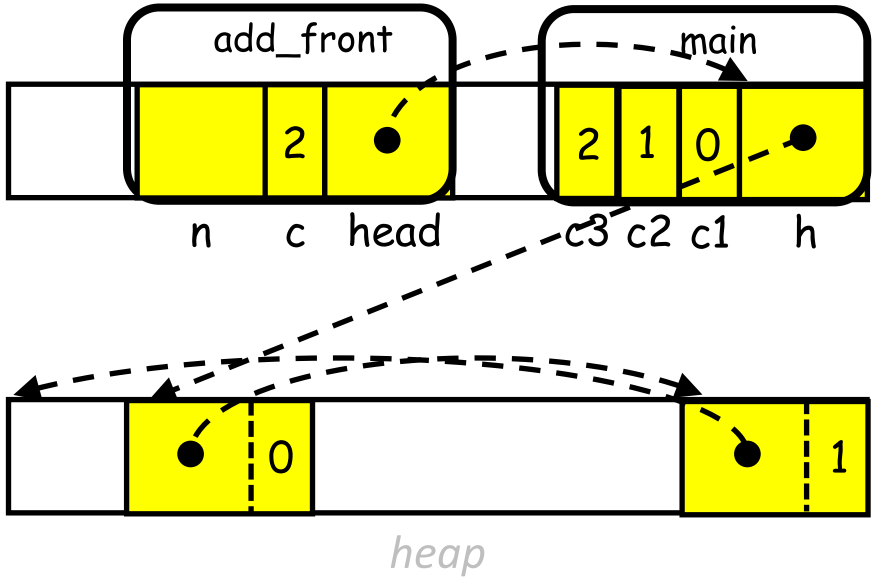
main.c

```
#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;
}
```

charList.c

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



Linked lists

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

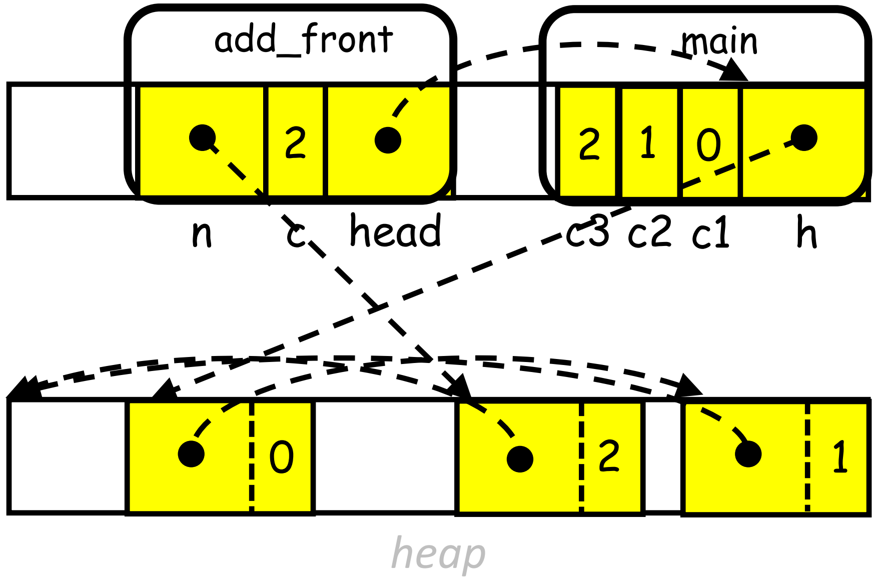
charList.c

```
#include "charList.h"
#include "assert.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

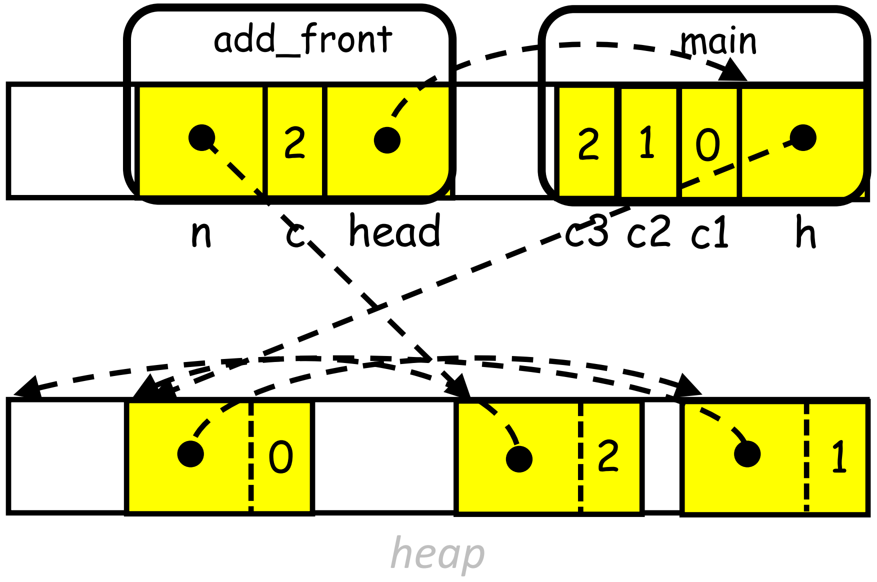
charList.c

```
#include "charList.h"
#include "assert.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

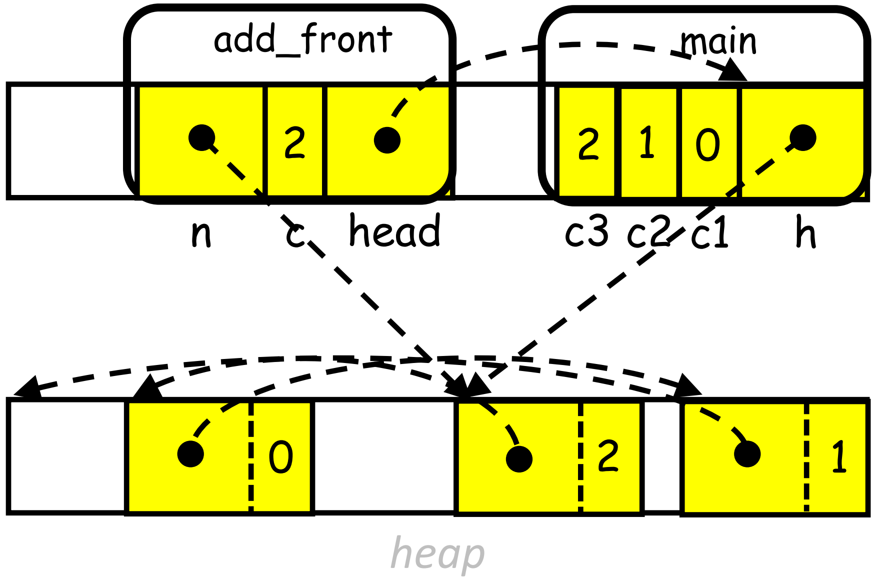
main.c

```
#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;
}
```

charList.c

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



Linked lists

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

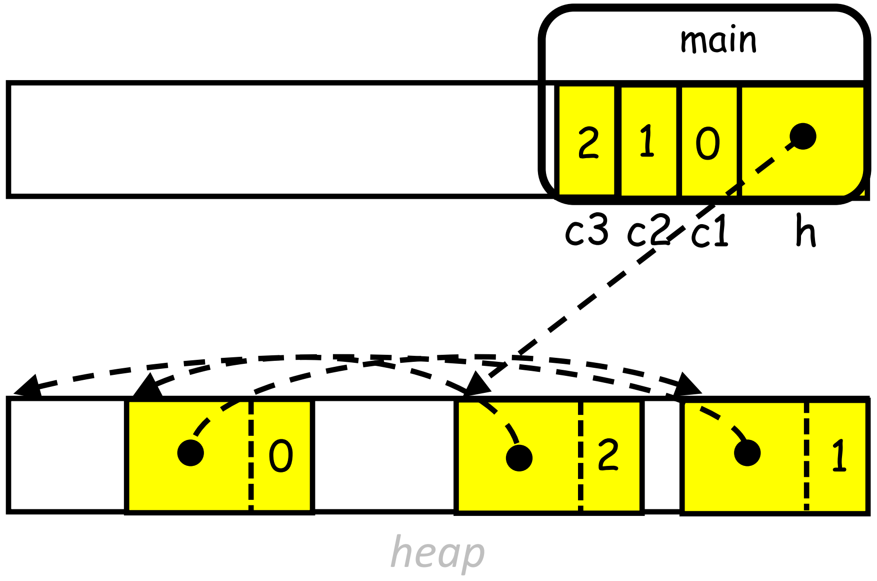
charList.c

```
#include "charList.h"
#include "assert.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;
}
```



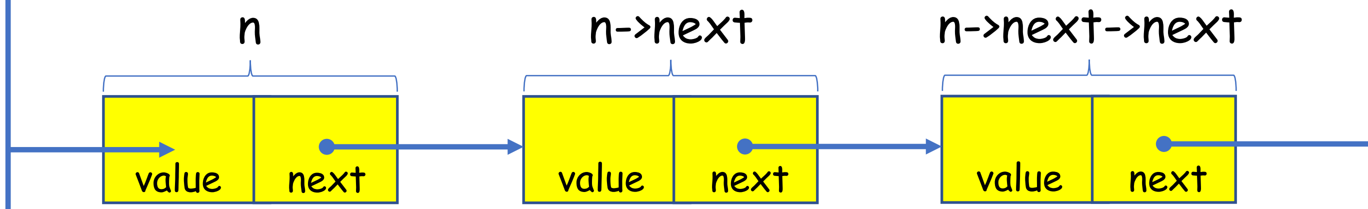
Linked lists

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

charList.c
```
#include "charList.h"
#include "assert.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
```



Linked lists

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

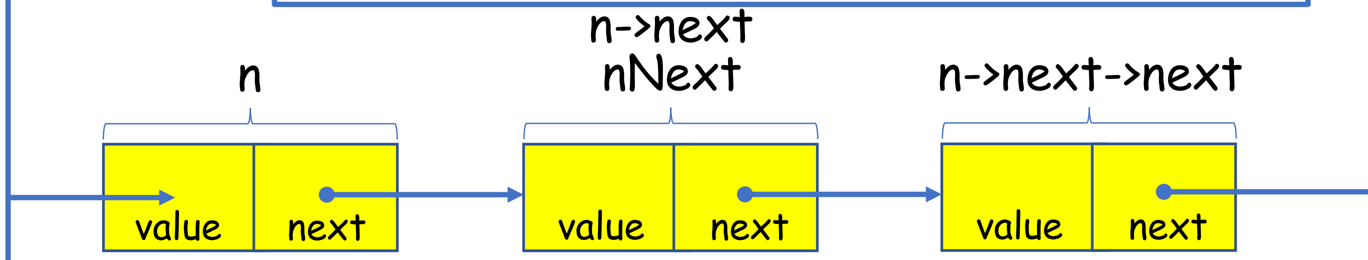
charList.h

```
#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 );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

charList.c

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



Linked lists

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

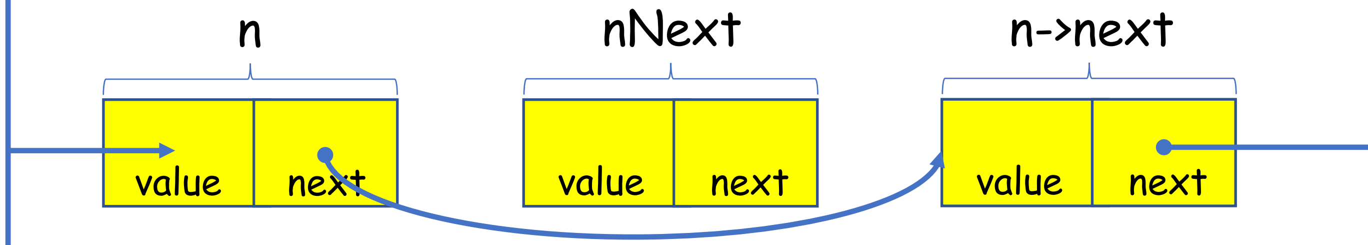
charList.h

```
#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 );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

charList.c

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



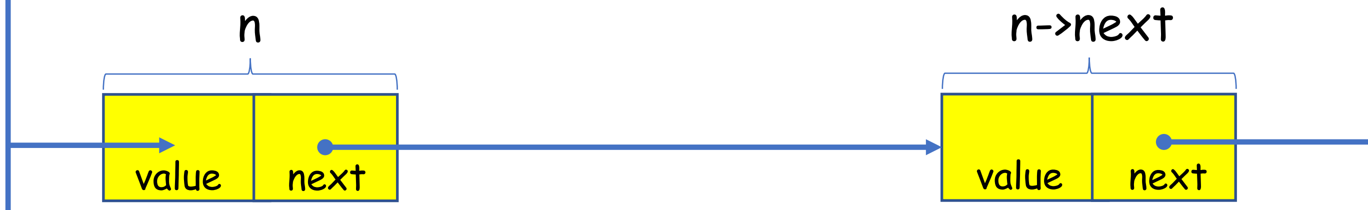
Linked lists

- 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 );
int length( const Node *head );
void print( const Node *head );
...
#endif // charList_included
```

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



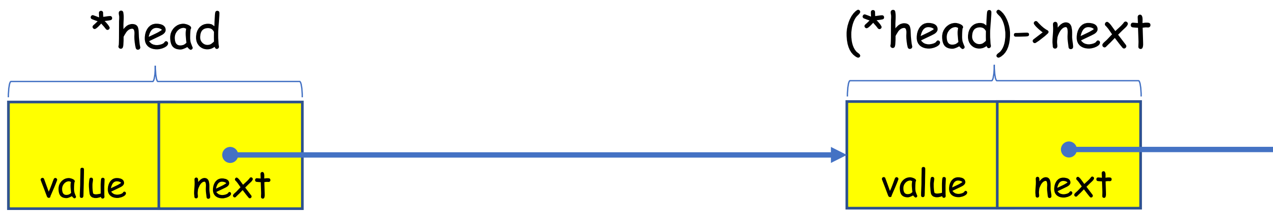
Linked lists

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

charList.c
```c
#include "charList.h"
#include "assert.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
```



Linked lists

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

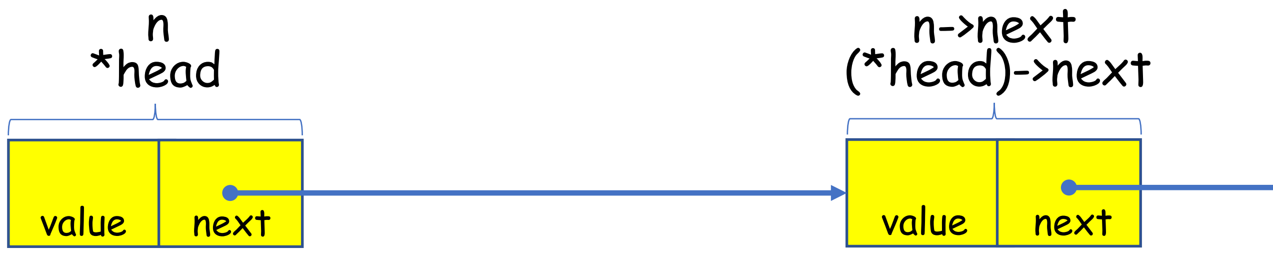
charList.h

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



Linked lists

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

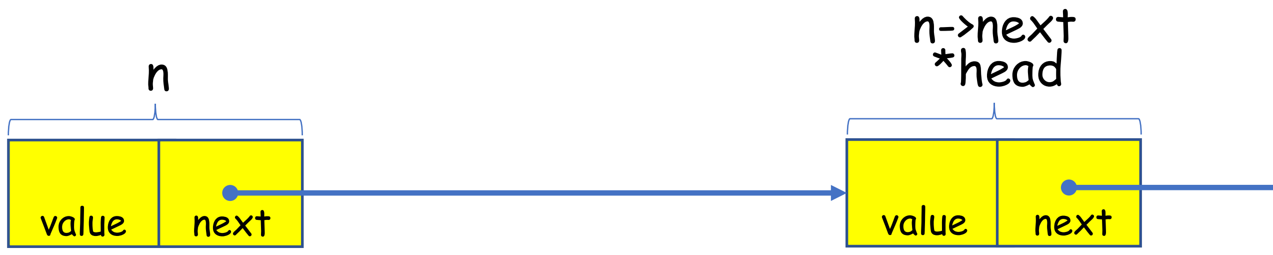
charList.c

```
#include "charList.h"
#include "assert.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
```



Linked lists

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

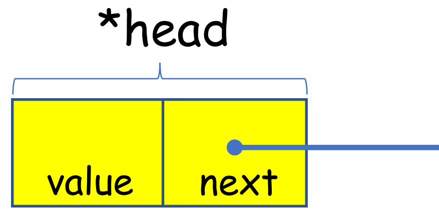
charList.c

```c
#include "charList.h"
#include "assert.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
```



Linked lists

Example (sorting chars)

- Read in chars from the stdin and insert them into a linked list, sorted from smallest to largest
    - Read the chars 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) chars
    - 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 );
#endif // charList_included
```

main.c

```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;
}
```

charList.h

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

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

main.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;
}
```

charList.h

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

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

head

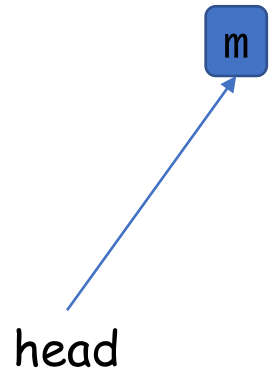
main.c

```
#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;
}
```

charList.h

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

```
>> ./a.out
m
```



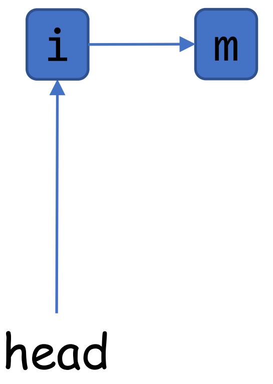
main.c

```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;
}
```

charList.h

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

```
>> ./a.out
mi
```



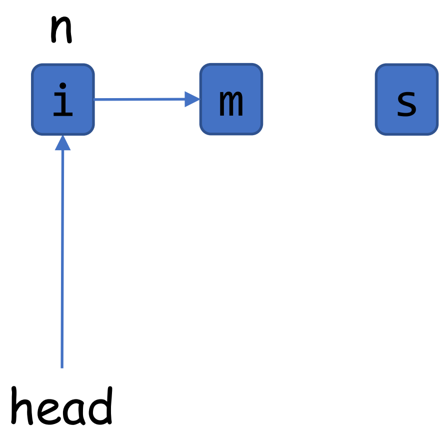
main.c

```
#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;
}
```

charList.h

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

```
>> ./a.out
mis
```



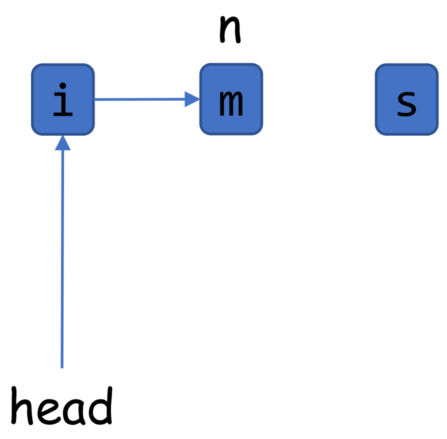
main.c

```
#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;
}
```

charList.h

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

```
>> ./a.out
mis
```



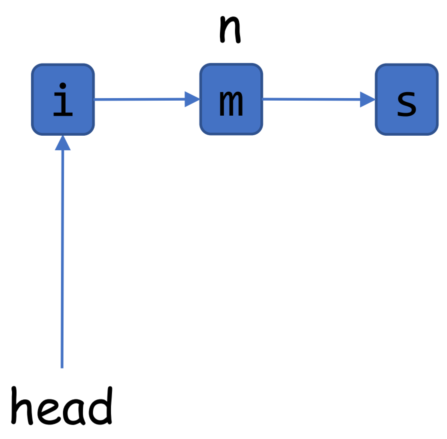
main.c

```
#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;
}
```

charList.h

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

```
>> ./a.out
mis
```



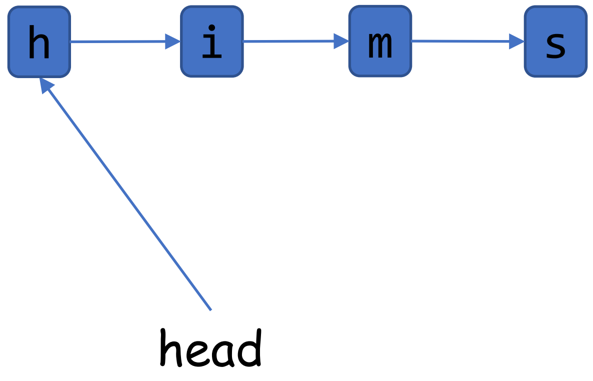
main.c

```
#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;
}
```

charList.h

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

```
>> ./a.out
mish
```



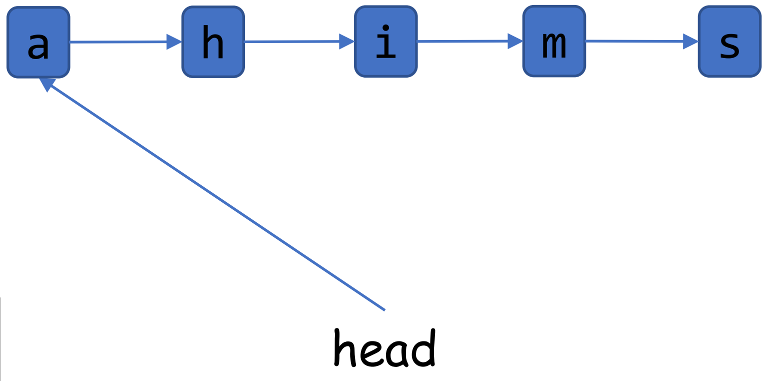
main.c

```
#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;
}
```

charList.h

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

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



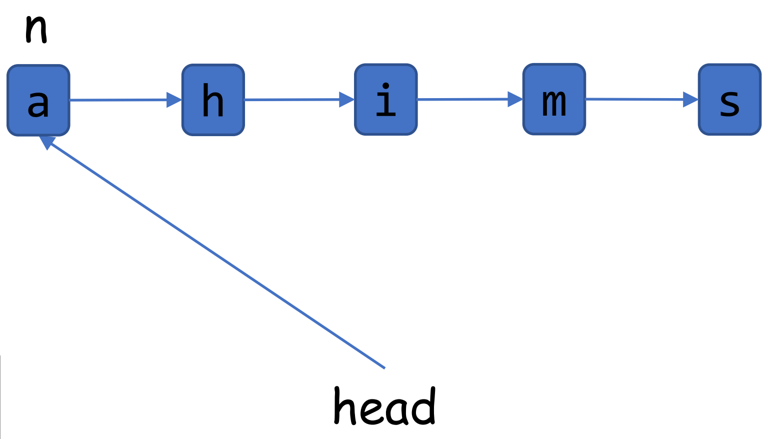
main.c

```
#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;
}
```

charList.h

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

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



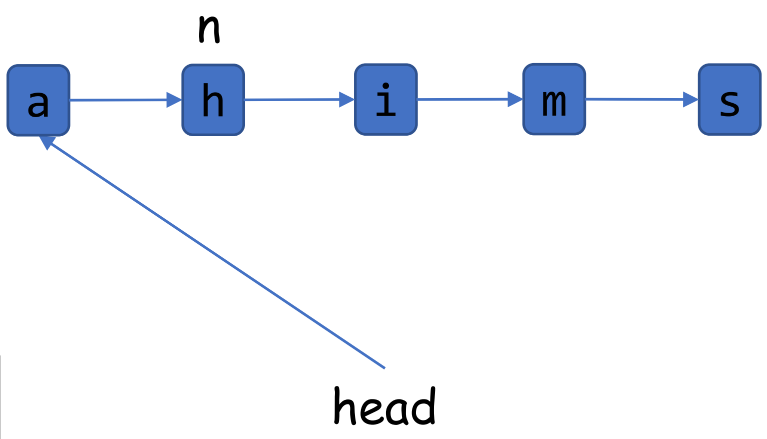
main.c

```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;
}
```

charList.h

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

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



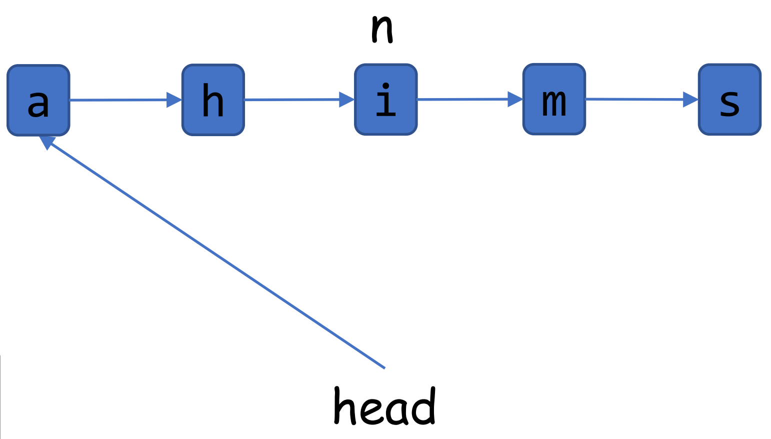
main.c

```
#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;
}
```

charList.h

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

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



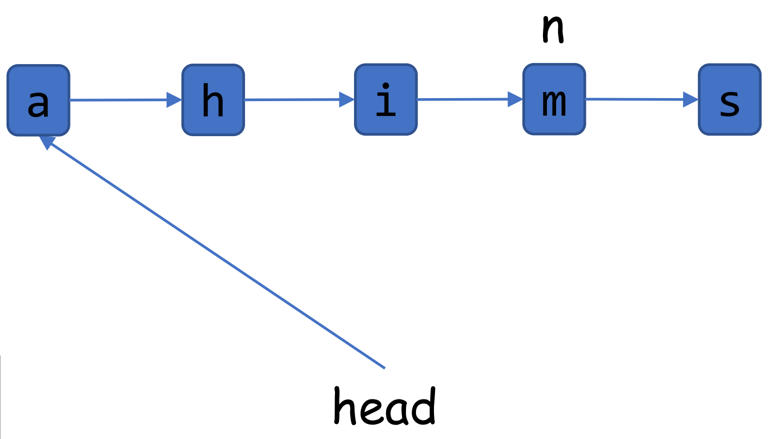
main.c

```
#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;
}
```

charList.h

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

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



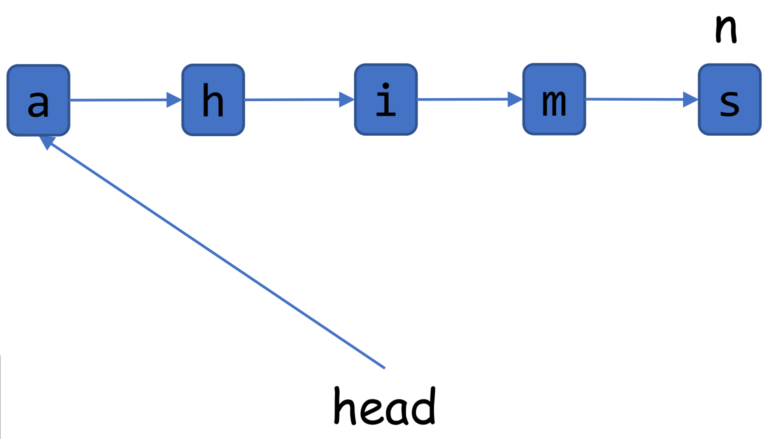
main.c

```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;
}
```

charList.h

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

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



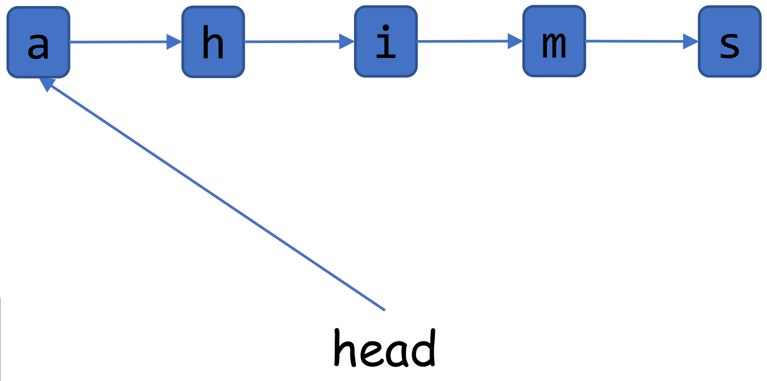
main.c

```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;
}
```

charList.h

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

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



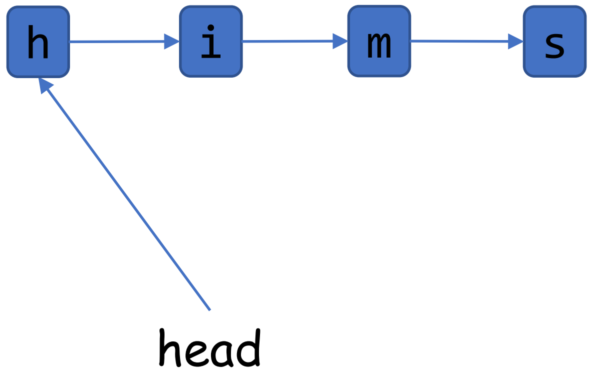
main.c

```
#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;
}
```

charList.h

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

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



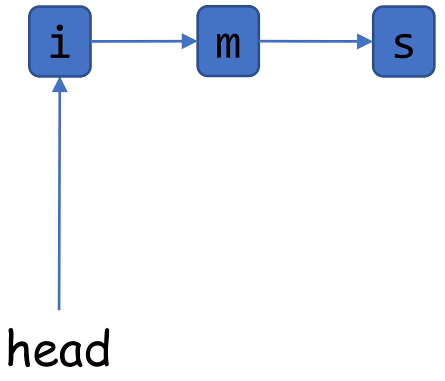
main.c

```
#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;
}
```

charList.h

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

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



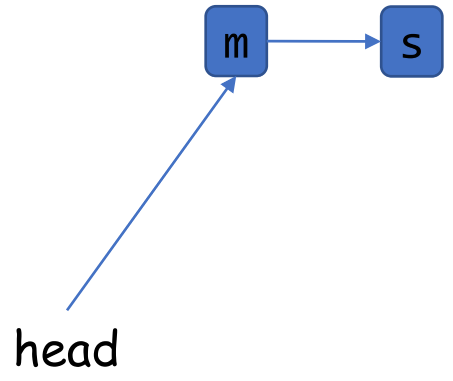
main.c

```
#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;
}
```

charList.h

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

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



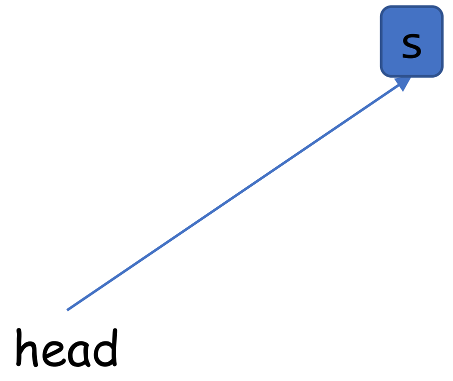
main.c

```
#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;
}
```

charList.h

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

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



main.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;
}
```

charList.h

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

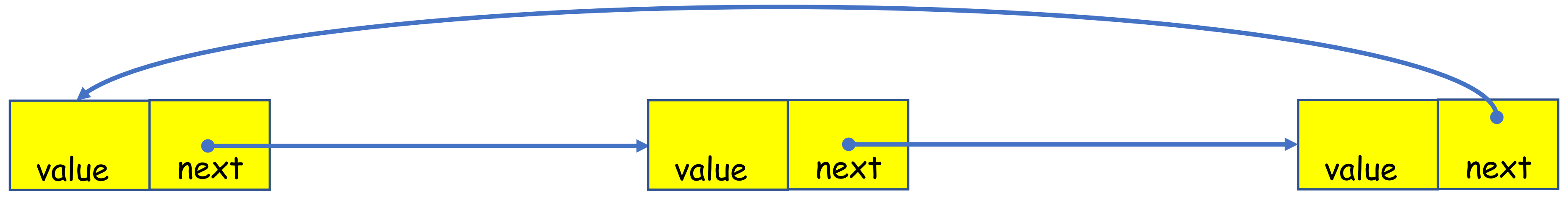
```
>> ./a.out
misha
ahims
>>
```

head

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
```

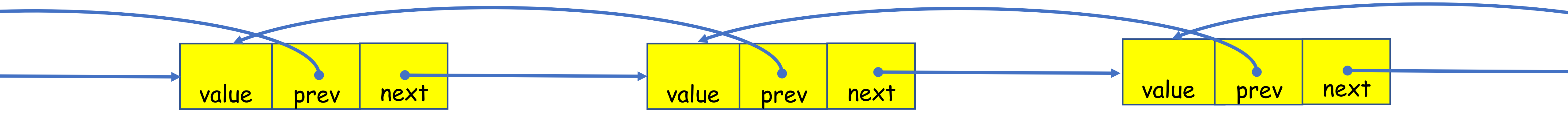


Linked lists

- 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
```



Outline

- Exercise 6-1
- Linked lists
- Review questions

Review questions

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

Review questions

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

Review questions

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

Review questions

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

Review questions

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

Exercise 6-1

- Website -> Course Materials -> ex6-2