## Today's plan

- Remainder:
    - Find your mid-term project partner and register your team on Piazza.
    - Homework 4 dues tomorrow (March 2nd).
- Review
- Ex 6-1

## Ex 5-3 - random integers

Which of the following generate a random integer between -5 and 8 inclusively?

A `rand() % 8 - 5`

B `rand() % 13 - 5`

C `rand() % 14 - 5`

D `rand() / RAND_MAX * 13 - 5`

E `(double)rand() / RAND_MAX * 13 - 5`

# Ex 5-3 - two's complement

Given a 8 bits `unsigned char` value, which of the following compute it's two's complement?

A `~value | 1;`

B `!value | 1;`

C `~value + 1;`

D `!value + 1;`

E `256 - value;`

## Ex 5-3 - little and big endian

If we use the below code to write an integer array to a binary file on a little endian machine (e.g. Linux)

```c
int data[5] = {1, 2, 3, 4, 5};
FILE *fp = fopen("wb");
fwrite(data, sizeof(int), 5, fp);
fclose();
```

Can we use the below code to read the array from the binary file on a big endian machine (e.g. Windows)?

```c
int data[5] = {0};
FILE *fp = fopen("rb");
fread(data, sizeof(int), 5, fp);
fclose();
```
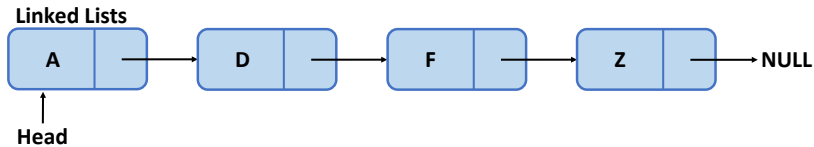
A Yes

B No

# Ex 5-3 - little and big endian

What does the below code do?

```
int data[5] = {1, 2, 3, 4, 5};
char* ptr = NULL;
for (int i = 0; i < 5; ++i) {
    ptr = (char*)&data[i];
    for (int j = 0; j < sizeof(int) / 2; ++j) {
        char temp = ptr[j];
        ptr[j] = ptr[sizeof(int) - j - 1];
        ptr[sizeof(int) - j - 1] = temp;
    }
}
```

# Linked lists

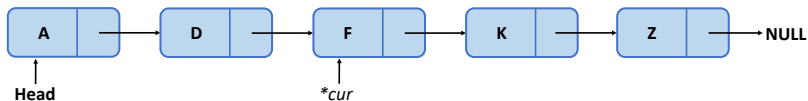Describe the linked list structure using a diagram.



How is the linked list's head different from a node?

# Linked lists

How do you calculate the `length` of a linked list?

length(*cur*);



A → D → F → K → Z → **NULL**

**Head**

*cur*

# Linked lists
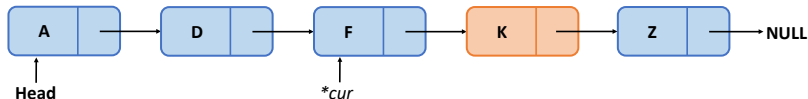
How do you implement the `add_after` function for a linked list?

add_after(*cur*, 'K');

1) Create a new node

K  →NULL

2) Add the new node after *cur*

A → D → F → K → Z →NULL

**Head**        *cur*

## Linked lists

Which of the following about arrays and linked lists is true?

  A Inserting new element into a linked list is faster than into an array.

  B We can dynamically change the size of an array but not a linked list.

  C Accessing the i'th element in a linked list is faster than accessing that in an array.

  D Removing a node in a linked list is easier than removing an element in an array.

  E Linked list requires more space than arrays.

## Linked lists

Consider the following program.

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct node_ {
    char data;
    struct node_ *next;
} Node;
int main(void) {
    Node *a = malloc(sizeof(Node)),
    *b = malloc(sizeof(Node)),
    *n;
    a->data = 'A';
    b->data = 'B';
    a->next = b;
    b->next = a;
    for (n = a; n != NULL; n = n->next) {
        printf("%c ", n->data);
    }
    printf("\n");
    return 0;
}
```

What output is printed?

A  No output is printed.

B  A

C  A B

D  B A

E  None of the above.

# Class exercises

Ex 6-1