Program memory: `free`; `struct`s

COSC 208, Introduction to Computer Systems, 2022-03-03

Announcements

- Project 1 Part B tonight

Warm-up

- Q1: *Describe the memory deallocation mistake for code snippet below?*

```
    int *ptrA = malloc(sizeof(int) * 3);
    int *ptrB = ptrA;
    free(ptrA);
    free(ptrB);
```

```
A.
```

```
int *ptr = malloc(sizeof(int) * 3);
ptr[0] = 1;
free(ptr);
ptr[1] = 2;
```

```
B.
```

```
int *ptr = malloc(sizeof(int) * 3);
ptr++;
free(ptr);
```

```
C.
```

```
int *ptrA = malloc(sizeof(int) * 3);
int *ptrB = ptrA;
ptrA[0] = 0;
ptrB[1] = 1;
free(ptrA);
ptrB[2] = 2;
```

```
D.
```

Pointers to structs

Assume you are given the following code:

```
struct account {
    int number; // Account number
    int balance; // Current account balance
};
int deposit(struct account *acct, int amount);
int transfer(struct account *from, struct amount *to, int amount);
```

Q2: *Write the* `deposit` *function, which adds* `amount` *to the balance of* `acct`*. The function should return the amount deposited.*

Q3: *Write the* `transfer` *function which moves* `amount` *from one account to another. The function should return the amount transferred if the transfer was successful or 0 otherwise.*

Dynamic DS practice

Two structs have been defined representing a queue and an item on a queue.

```
struct item {
    int value;
    struct item *next;
};
struct queue {
    struct item *head;
    struct item *tail;
};
```

The new_queue *function creates a new, empty queue.*

```
struct queue *new_queue() {
    struct queue *q = malloc(sizeof(struct queue));
    q->head = NULL;
    q->tail = NULL;
    return q;
}
```

Q4: *Write a function called* enqueue *that adds a new value at the end of the queue.*

Q5: *Write a function called* dequeue *that removes and returns the value at the head of the queue. The function should return -1 if the queue is empty.*

Q6: *Write a function called* `free_queue` *that empties and frees a queue.*