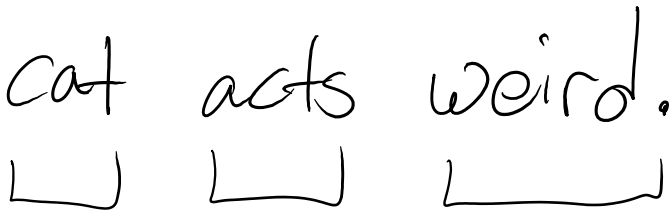
October 3, 2017

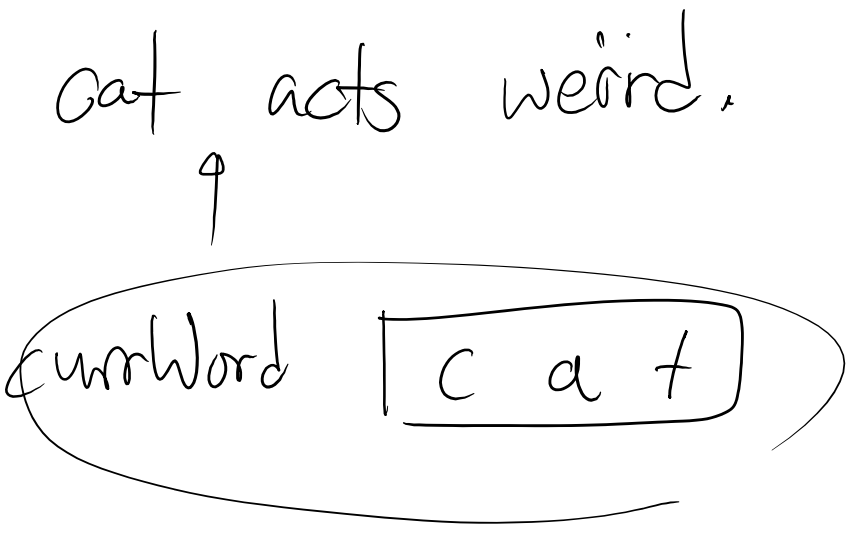
October 3, 2017 11:03 AM



cat => 99+97+116 => 312

acts => 97+99+116+115

weird. => 119+101+105+114+100+46

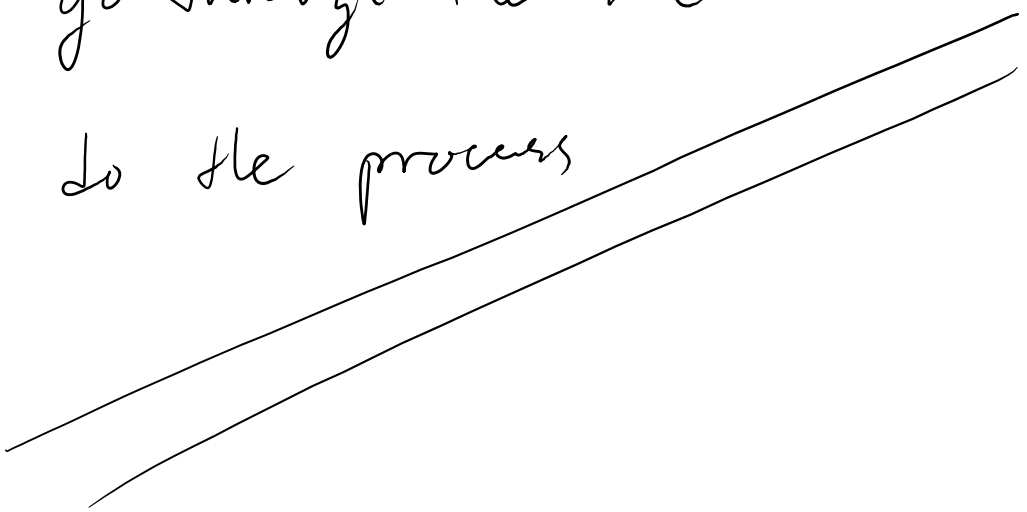



```
#include <stdio.h>
```

readline (stdin)

go through the line

do the process



- #define LINE_MAX ??
- fgets() ⇒ '\n' at the end
- contacts weird.

to prevent buffer overrun

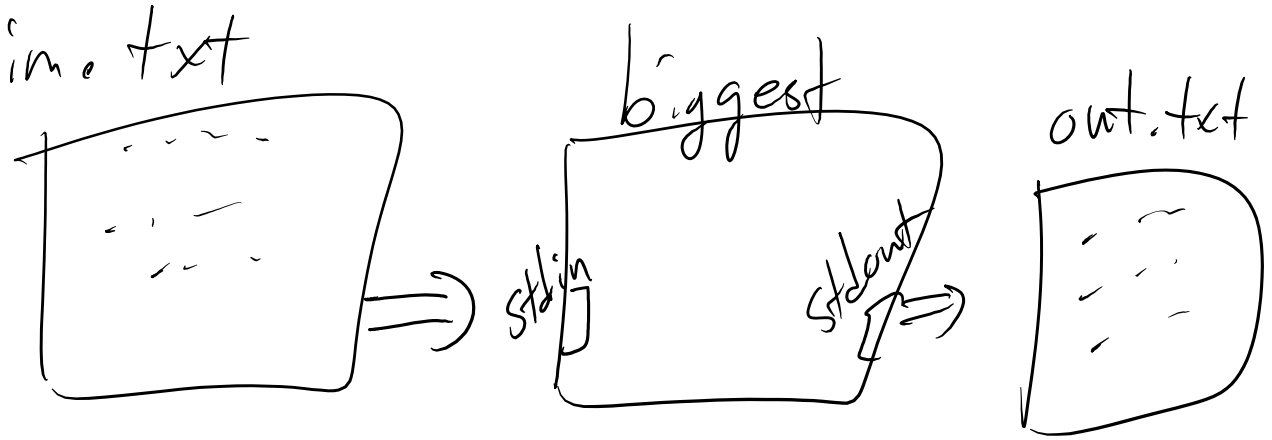
make word array to be same size with the line array

```
char line[LINE_MAX];
char currWord[LINE_MAX];
```

```
fgets(line, LINE_MAX, stdin)
printf()
```

```
clang -Wall biggest.c -o biggest
```

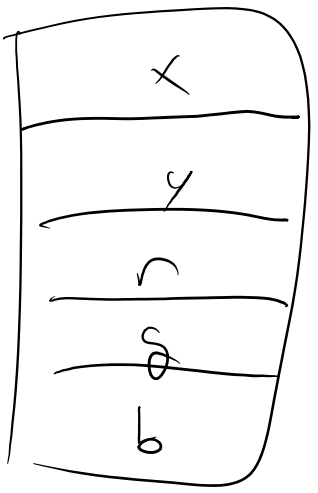
./biggest < in.txt > out.txt




```
struct Color {
    char r;
    char g;
    char b;
};
```

```
struct Pixel {
    int x;
    int y;
    struct {
        char r;
        char g;
        char b;
    } color;
};
```

```
struct Pixel p;
p.color.r;
```




```
struct Color {
    ....
};

struct Pixel {
    int x;
    int y;
```

```
    int y;
    struct Color color;
};
```

```
struct Node {
    int data;
    struct Node* next;
};
typedef struct Node Node;

Node *head = NULL;

int foo() {
    Node node;
```

Node node;

head = &node;

malloc

char* str = "hello";

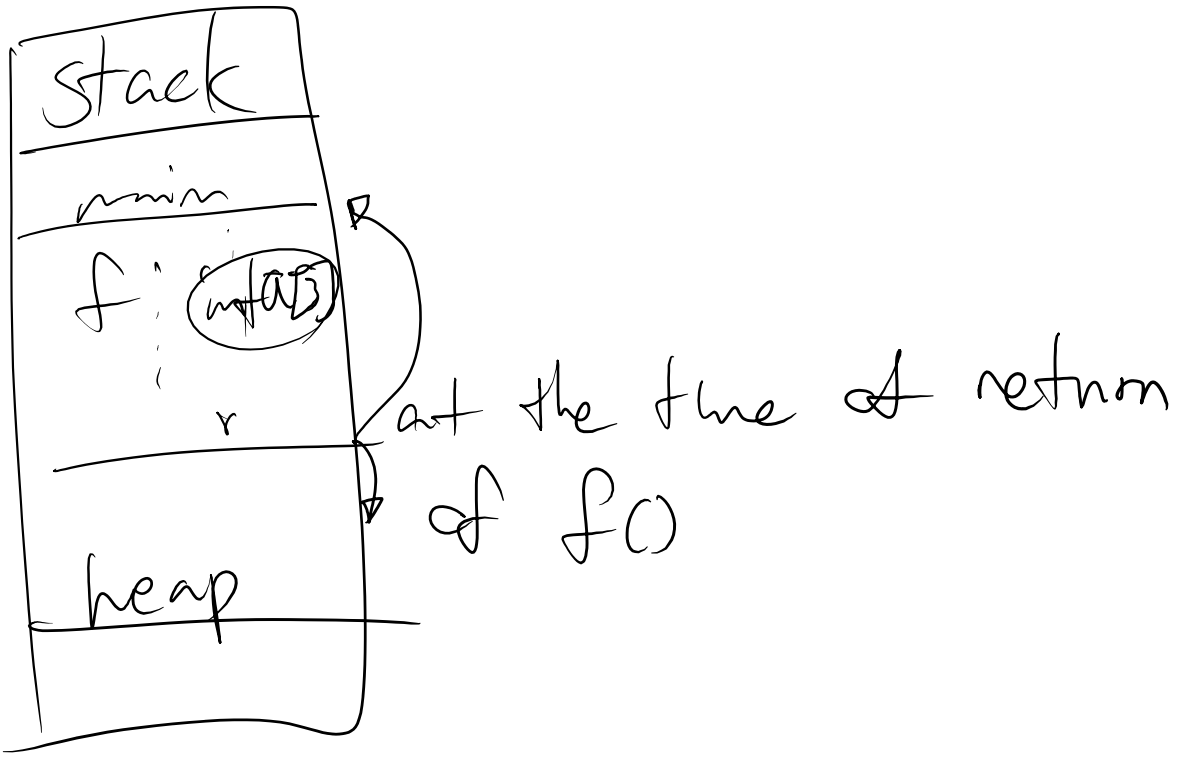
char str[] = "hello";

&

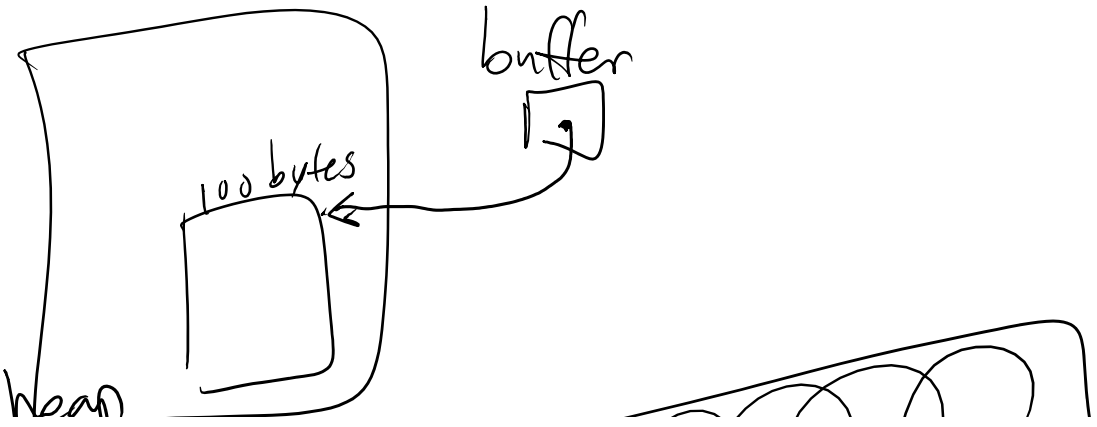
RTS

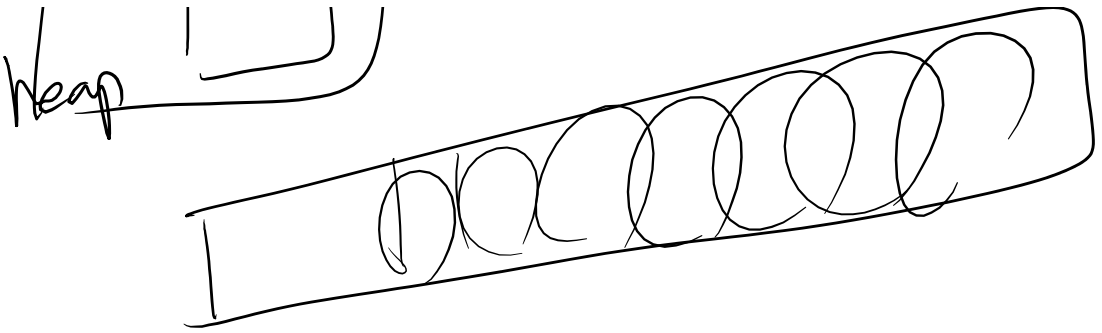
```
int main() {
    int x;
```

int y;

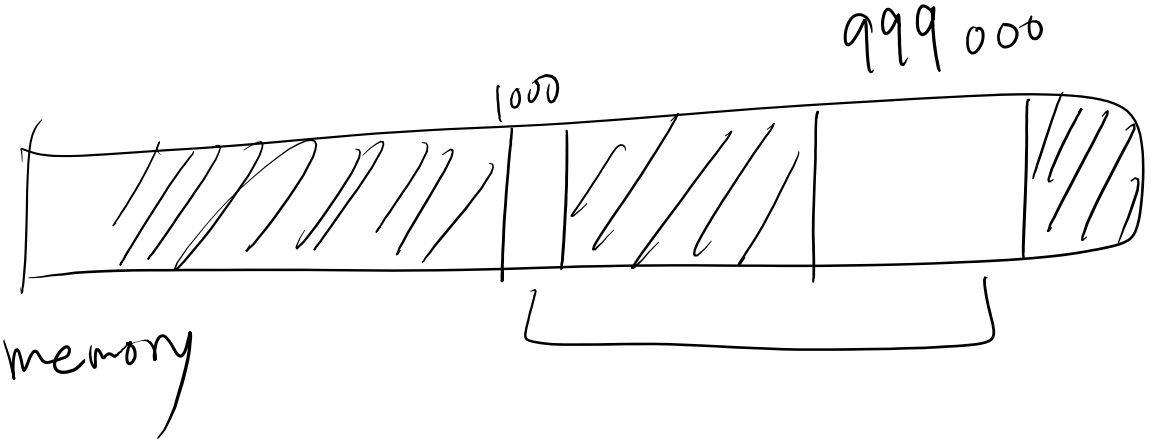


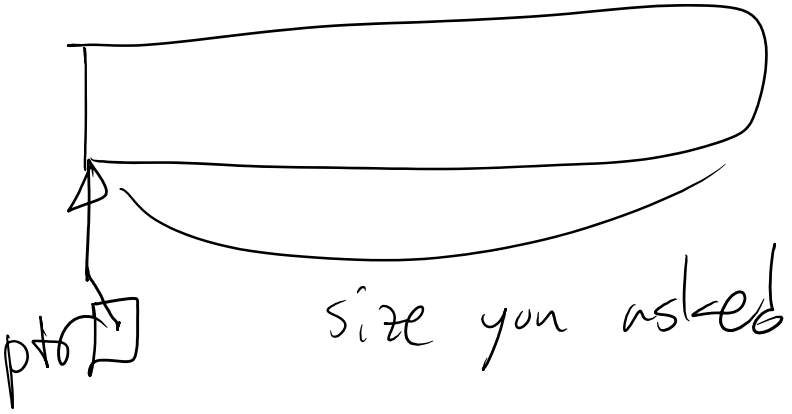

```
#include <stdlib.h>

int main() {
    char* buffer = (char*) malloc(100);
```






```
void* ptr = malloc(1000000);

if(NULL == ptr)
    // failed
```





```
Node* head = NULL;

void add (int data) {
    Node* newNode = (Node*)
        malloc (sizeof(Node));
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}
```

Java:

```
node.next = null;
```

G.C.

```
free( void* )
```

```
void removeFirst() {
    Node* old = head;
    head = head->next;
    free(old);
}
```

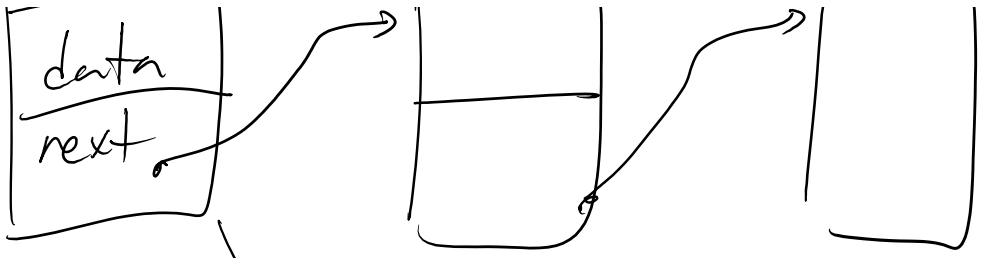
good

```
free(head);
head = head->next;
// work? null? seg fault?
```

bad!

head = NULL;

without free() our node is MIA



free() will not free the next

---

if you allocate, you free them

```
void clear() {
    Node * curr = head;
    Node * next = NULL;
    while (NULL != curr) {
        next = curr->next;
        free(curr); //order...
        curr = next;
    }
    head = NULL;
```

```
    head = NULL;
}
```

```
char* ptr = "hello";
free(ptr); // bad
```

```
char * ptr = (char*) malloc(10);
free(ptr);
free(ptr); // bad
if(NULL != ptr) {
    free(ptr);
```

```
    ptr = NULL;
}

char* strdup(...) { }
```