CPS125
with Prof. Kosta Derpanis

Final exam review

Part 2
Administration
Final Exam Info:
Topics include the material covered after the midterm but knowledge of the basics of C programming is expected. The focus of the exam is on lessons 6, 7, 8 but knowledge of lessons 2 to 5 is expected. No binary conversion questions (lesson 1). Knowledge of labs 5 to 10 is expected as well as project #2. The exam consists of 30 multiple choice questions and 3 programming questions. The duration is 2 hours. Because lesson 8 is not associated with any labs, that lesson will be evaluated by multiple choice questions only in the exam.
Final Exam Info:
Topics include the material covered after the midterm but knowledge of the basics of C programming is expected. The focus of the exam is on lessons 6, 7, 8 but knowledge of lessons 2 to 5 is expected. No binary conversion questions (lesson 1). Knowledge of labs 5 to 10 is expected as well as project #2. The exam consists of 30 multiple choice questions and 3 programming questions. The duration is 2 hours. Because lesson 8 is not associated with any labs, that lesson will be evaluated by multiple choice questions only in the exam.
Write a void function that takes in a 2D array of integers and returns, using three pointers, the number of zero, positive and negative array elements.

Assume the 2D array has 10 columns.
Problem SOLVING
Problem definition
Problem definition

Algorithm
Problem definition
Algorithm
Implementation
Write a void function that takes in a 2D array of integers and returns, using three pointers, the number of zero, positive and negative array elements.

Assume the 2D array has 10 columns.
```c
void myCount(int a[][10], int numRows, int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos  = 0;
    *numNeg  = 0;

    for (row = 0; row < numRows; row++){
        for (col = 0; col < 10; col++){
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if(a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
```
void myCount(int a[][10], int numRows, int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos  = 0;
    *numNeg  = 0;

    for (row = 0; row < numRows; row++){
        for (col = 0; col < 10; col++){
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if(a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
void myCount(int a[][10], int numRows, int *numZero, int *numPos, int *numNeg) {
    int row, cols;

    *numZero = 0;
    *numPos = 0;
    *numNeg = 0;

    for (row = 0; row < numRows; row++) {
        for (col = 0; col < 10; col++) {
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if (a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
void myCount(int a[][10], int numRows,
    int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos  = 0;
    *numNeg  = 0;

    for (row = 0; row < numRows; row++){
        for (col = 0; col < 10; col++){
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if(a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
void myCount(int a[][10], int numRows, int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos = 0;
    *numNeg = 0;

    for (row = 0; row < numRows; row++){
        for (col = 0; col < 10; col++){
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if(a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
void myCount(int a[][10], int numRows, int *numZero, int *numPos, int *numNeg)
{
    int row, cols;

    *numZero = 0;
    *numPos = 0;
    *numNeg = 0;

    for (row = 0; row < numRows; row++)
    {
        for (col = 0; col < 10; col++)
        {
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if (a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
void myCount(int a[][10], int numRows, int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos  = 0;
    *numNeg  = 0;

    for (row = 0; row < numRows; row++){
        for (col = 0; col < 10; col++){
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if(a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
void myCount(int a[][10], int numRows, 
    int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos  = 0;
    *numNeg  = 0;

    for (row = 0; row < numRows; row++)
        for (col = 0; col < 10; col++)
            if (a[row][col] == 0) 
                *numZero = *numZero + 1;
            else if(a[row][col] > 0) 
                *numPos = *numPos + 1;
            else 
                *numNeg = *numNeg + 1;
}

void myCount(int a[][10], int numRows,
int *numZero, int *numPos, int *numNeg){
    int row, cols;

    *numZero = 0;
    *numPos  = 0;
    *numNeg  = 0;

    for (row = 0; row < numRows; row++){
        for (col = 0; col < 10; col++){
            if (a[row][col] == 0)
                *numZero = *numZero + 1;
            else if(a[row][col] > 0)
                *numPos = *numPos + 1;
            else
                *numNeg = *numNeg + 1;
        }
    }
}
STRINGS
No native string type in C
No native string type in C

To represent strings, use an array of `chars`
No native string type in C

To represent strings, use an array of `chars`

A string is an array of characters ending with `\0`
No native string type in C

To represent strings, use an array of `char`

A string is an array of characters ending with the null character `\0`
No native string type in C

To represent strings, use an array of `char`

A string is an array of characters ending with a null character (\0)
char city[] = {'T', 'o', 'r', 'o', 'n', 't', 'o', '\0'};
char city[] = {'T','o','r','o','n','t','o','\0'};
char city[] = {'T','o','r','o','n','t','o','\0'};

equivalent

cchar city[] = “Toronto”;

T 0
O 1
r 2
o 3
n 4
t 5
o 6
\0 7
single quotes vs. double quotes
single character

single quotes vs. double quotes
single character

single quotes vs. double quotes
single character

single quotes vs. double quotes

string

'c'
single quotes vs. double quotes

single character

c

string

"Ryerson"
single character

single quotes vs. double quotes

string

'Ryerson'

‘c’

“c”
String size equals number of characters plus one
String size equals number of characters plus one end-of-string character.
String size equals number of characters plus one end-of-string character
Linked list
linked list

an ordered set of data elements, each containing a link to its successor
linked list

an ordered set of data elements, each containing a link to its successor

and sometimes its predecessor
start

1 -> 2 -> 3 (X)
start

structure

1 -> 2 -> 3

×
start

structure

data fields
start

1

structure

data fields

address to next element

2

3

×
typedef struct element {
    int num;
    struct element* next;
} element;

int main (void) {
    element e1, e2, e3, e4, *current;
    e1.num = 1;
    e2.num = 2;
    e3.num = 3;
    e4.num = 4;
    current = &e2;
    e1.next = NULL;
    e2.next = &e3;
    e3.next = &e4;
    e4.next = &e1;

    while (current != NULL) {
        printf (%d, current->num);
        current = current->next;
    }
    return 0;
}
int main (void) {
    element e1, e2, e3, e4, *current;
    e1.num = 1;
    e2.num = 2;
    e3.num = 3;
    e4.num = 4;
    current = &e2;
    e1.next = NULL;
    e2.next = &e3;
    e3.next = &e4;
    e4.next = &e1;

    while (current != NULL) {
        printf ("%d", current->num);
        current = current->next;
    }
}

Console
2341
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;

e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
element e1, e2, e3, e4, *current;
e1.num = 1;
e2.num = 2;
e3.num = 3;
e4.num = 4;
current = &e2;
e1.next = NULL;
e2.next = &e3;
e3.next = &e4;
e4.next = &e1;
typedef struct elephant
{
    char name[10];
    int weight;
    struct elephant* next;
}elephant;
typedef struct elephant 
{
    char name[10];
    int weight;
    struct elephant* next;
} elephant,
elephant *current, *first;
int response;

/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response);

while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
current->next = 
    */
elephant *current, *first;
int response;

/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response);

while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
current->next =
    current = current->next;
}
current->next = NULL;
elephant *current, *first;
int response;

/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response);

while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
    current = current->next;
    printf("Elephant name? ");
    scanf ("%s", current->name);
    printf("Elephant weight? ");
    scanf ("%d", &current->weight);
    printf("\nAdd another? (y=1/n=0) ");
    scanf ("%d", &response);
    current->next = NULL;"
elephant *current, *first;
int response;

/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf ("%d", &response);

while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next = (elephant *)calloc(1,sizeof(elephant));
current = current->next;
    printf("Elephant name? ");
    scanf ("%s", current->name);
    printf("Elephant weight? ");
    scanf ("%d", &current->weight);
    printf("\nAdd another? (y=1/n=0)");
    scanf ("%d", &response);
}
/ * create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf("%d", &response);

while (response) /*while response is 1 (yes) */ {
    /* allocate node and change current pointer */
current->next =
    (elephant *)calloc(1,sizeof(elephant));
current = current->next;

    /* fill node */
    printf("Elephant name? ");
    scanf("%s", current->name);
    printf("Elephant weight? ");
    scanf("%d", &current->weight);
}
current->next = NULL;
printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response)

while (response) /*while response is 1 (yes) */ {
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
current = current->next;

    /* fill node */
    printf("Elephant name? ");
    scanf ("%s", current->name);
    printf("Elephant weight? ");
    scanf ("%d", &current->weight);

    printf("\nAdd another? (y=1/n=0) ");
    scanf ("%d", &response);
}
current->next = NULL;
```c
int response;

/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf("%s", current->name);

printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)"ız);
scanf("%d", &response);

while (response) /*while response is 1 (yes)*/ {
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
current = current->next;

    /* fill node */
    printf("Elephant name? ");
    scanf("%s", current->name);
    printf("Elephant weight? ");
    scanf("%d", &current->weight);

    printf("\nAdd another? (y=1/n=0)"ız);
    scanf("%d", &response);
}
current->next = NULL;
```
/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");  
scanf ("%d", &response);
}
current->next = NULL;
current->next = NULL;

set last node’s successor link to NULL
elephant *current, *first;
int response;
elephant *current, *first;
int response;
elephant *current, *first;
int response;
elephant *current, *first;
int response;

first

current
elephant *current, *first;
int response;
/ create first node /

first = (elephant*)calloc(1,sizeof(elephant));
current = first;
/* create first node */

```c
first = (elephant*)calloc(1,sizeof(elephant));
current = first;
```
/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;
/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;
`/* create first node */

first = (elephant*)calloc(1,sizeof(elephant));
current = first;`
/* create first node */
first = (elephant*)calloc(1,sizeof(elephant));
current = first;
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf ("%d", &response)
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf ("%d", &response)
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf("%d", &response)
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response)
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);
printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response)
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf ("%d", &response)

Edna
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response)

- first
  "Edna" 450
- current

Edna
450
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response)
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response)
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf ("%d", &response)

user input: 1
while (response) /*while response is 1 (yes) */ {
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;
}
while (response) /*while response is 1 (yes) */ {
    /* allocate node and change current pointer */
    current->next =
    (elephant *)calloc(1,sizeof(elephant));
    current = current->next;
}
```c
while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1, sizeof(elephant));
    current = current->next;
}
```

first

```
"Edna" 450
```

current
while (response) /*while response is 1 (yes) */ {
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;
}
while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
current->next =
    (elephant *)calloc(1,sizeof(elephant));
current = current->next;

"Edna" 450 current

first

...
while (response) /*while response is 1 (yes) */ {
   /* allocate node and change current pointer */
   current->next =
       (elephant *)calloc(1,sizeof(elephant));
   current = current->next;
}
while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;

    "Edna" 450

/* fill node */

printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf("%d", &response);

/* fill node */

printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf("%d", &response);
/* fill node */

printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)"); scanf ("%d", &response);
}
/ * fill node *

printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response);
/* fill node */
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)"); scanf ("%d", &response);
/ * fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf("%d", &response);

first

![First node diagram]

current

![Current node diagram]
/* fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf("%d", &response);
}
while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;

    first

    current

    "Edna" 450

    "Elm" 600
while (response) /*while response is 1 (yes)*/ {
    /* allocate node and change current pointer*/
    current->next =
      (elephant *)calloc(1,sizeof(elephant));
    current = current->next;

    "Edna" 450
    "Elm" 600
while (response) /*while response is 1 (yes) */ {
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;
}

```
---

```

```
Edna  450

Elm  600
---

```
while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
current->next = 
    (elephant *)calloc(1,sizeof(elephant));
current = current->next;

first

```
| "Edna" | 450 |
```

current

```
| "Elm"  | 600 |
```


while (response) /* while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;
}
while (response) /* while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1, sizeof(elephant));
    current = current->next;

    "Edna" 450
    "Elm" 600
while (response) /*while response is 1 (yes) */
{
    /* allocate node and change current pointer */
    current->next =
        (elephant *)calloc(1,sizeof(elephant));
    current = current->next;
}

first

current

“Edna”  450

“Elm”   600
/* fill node */

printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf ("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response);

/ * fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)" );
scanf("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf ("%s", current->name);
printf("Elephant weight? ");
scanf ("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf ("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf("%d", &response);
/* fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response);
/* fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response);
}
/* fill node */
printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0) ");
scanf("%d", &response);
/* fill node */

printf("Elephant name? ");
scanf("%s", current->name);
printf("Elephant weight? ");
scanf("%d", &current->weight);

printf("\nAdd another? (y=1/n=0)");
scanf("%d", &response);

user input: 0
```c
current->next = NULL;
```
current->next = NULL;
current->next = NULL;
for(current = first; current != NULL; 
current = current->next)
{
    printf("Name: %s, weight: %d\n, 
    current->name, current->weight);
}

"Edna" 450 "Elm" 600 "Elle" 475
for (current = first; current != NULL; 
    current = current->next) 
{
    printf("Name: %s, weight: %d\n, 
           current->name, current->weight);
}
for (current = first; current != NULL; 
    current = current->next) 
{
    printf("Name: %s, weight: %d
, 
        current->name, current->weight);
}
for (current = first; current != NULL; 
current = current->next)  
{
    printf("Name: %s, weight: %d
, 
current->name, current->weight);
}
for(current = first; current != NULL; 
current = current->next)
{
    printf("Name: %s, weight: %d\n,
            current->name, current->weight);
}
for (current = first; current != NULL; current = current->next) {
    printf("Name: %s, weight: %d\n", current->name, current->weight);
}
for(first = current; current != NULL; current = current->next)
{
    printf("Name: %s, weight: %d\n,
            current->name, current->weight);
}
for (current = first; current != NULL; current = current->next) {
    printf("Name: %s, weight: %d\n, current->name, current->weight); 
}
for (current = first; current != NULL; 
    current = current->next) 
{
    printf("Name: %s, weight: %d\n, 
          current->name, current->weight);
}
for (current = first; current != NULL; 
current = current->next)
{
    printf("Name: %s, weight: %d\n, 
current->name, current->weight);
}
for (current = first; current != NULL; current = current->next) {
    printf("Name: %s, weight: %d\n, current->name, current->weight);
}
for(current = first; current != NULL; 
    current = current->next)
{
    printf("Name: %s, weight: %d\n,
            current->name, current->weight);
}
for (current = first; current != NULL; 
    current = current->next)
{
    printf("Name: %s, weight: %d\n, 
            current->name, current->weight);
}
Things to know

strlen
fgets
strcpy
calloc
malloc
sizeof
typedef
struct
atoi
atof
Thank you