BE - Discussion 13, 2020-12-04

Announcements

- Exam 3 today from 4^{30}-6^{00} pm (EST) with 15 more minutes for uploading
  - → tomorrow at 7^{00} am (EST) for + time zones
  - → Becareful on the exam!
  - → you can use C but we truly don't recommend it... time waste!
  - → programs must follow good programming style but don't have to write comments unless specified by the problem
  - → don't make things harder than they are!!
  - → write general code
  - → Topics: All of the C topics covered
- Next Monday: in the morning, time for you to do Course Evals, and there will be Open Hours
  - → Lab attendance is required... it will be optional project presentations
- Final Project: due next Wednesday, 2020-12-09, at 9 am (EST) for EVERYONE!!!
  - → upload one document per group to Gradescope
  - → please make sure all group member names are on it

Review Quiz 8

Review of Material

- Shoutout what you want us to review! Otherwise, we'll go through Sample Exam and/or Study Session solutions
  - → pointers
  - → linked lists
  - → general math operations
  - → don't forget generating random integers!

Generating Random Integers General Formula: `rand()%(MAX - MIN + 1) + MIN`

put this on a sticky note right next to your computer!

```c
typedef struct linked_list
{
        char data;
        struct linked_list *next;
}       element;
typedef element *elementptr;
```


```c
void trav_print(elementptr);
```

← function prototype

```c
int main()
{
        elementptr first = NULL,
                   last = NULL;
        int i;
        char myword[20] = "Lake Tahoe";

(1)     first = (elementptr) malloc(sizeof(element));
        last = first;
        first -> data = myword[0];

(2)     for (i = 1; i < strlen(myword); i++)
        {
(2a)        last -> next = (elementptr) malloc(sizeof(element));
(2b)        last = last -> next;
(2c)        last -> data = myword[i];
        }

(3)     last -> next = NULL;
        trav_print(first);

        free(first);
        free(last);

        return 0;
}
```

initializations + Declarations

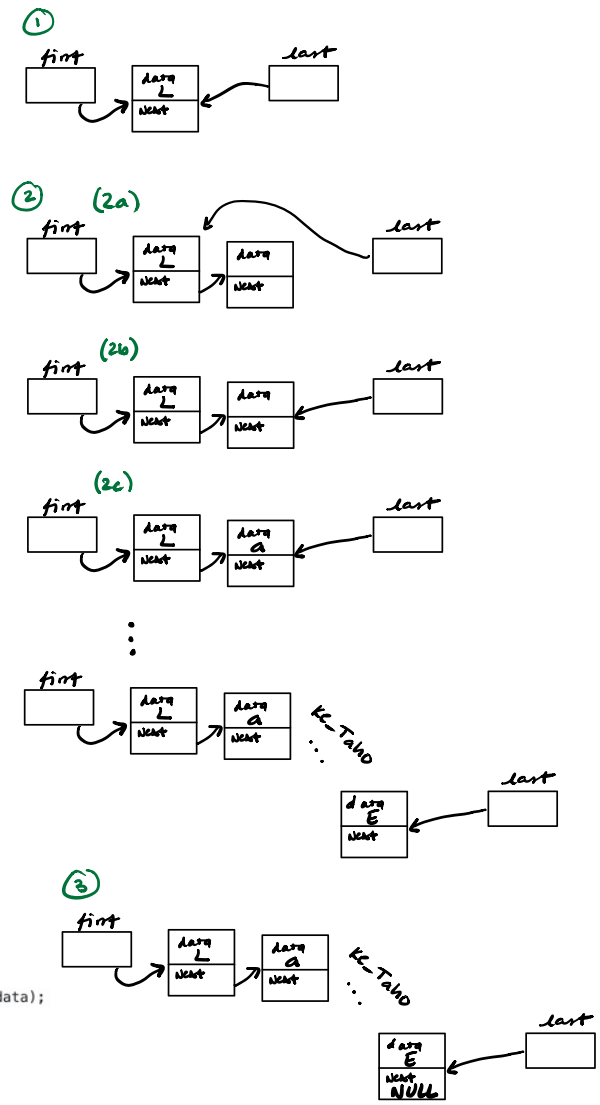
```c
void trav_print(elementptr fir)
{
        elementptr current;

        current = fir;
        if (current == NULL)
                printf("There is no linked list!\n");
        else
        {
                while (current != NULL)
                {
                        printf("The data value is %c\n", current -> data);
                        current = current -> next;
                }
        }
}
```

← function header

init/dec.

Cond.

Example of Creating a LL and traversing:



Output:

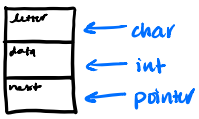
```
The data value is L
The data value is a
                  k
                  e
                  _
                  T
                  a
                  h
                  o
                  e
The data value is e
```

← space does show up!

Question: in function header, could we do first instead of fir? yes!

```c
// Define type & create linked list with a struct for each element and pointer
typedef struct linked_list
{
        char letter;
        int data;
        struct linked_list *next;
} element;

typedef element * elementptr;

// Function prototype for the trav_print function
void trav_print(elementptr);

int main()
{
        // Inititalization & Declaration here
        elementptr first = NULL,
        current, temp,
        last = NULL;

        char fname[10];

        int i;

        printf("\nLet's see an example of linked lists in action!\nReady?\n");
        printf("\n***\n\nThis is the beginning of the program.\n\n***\n");

        // Use strcpy to copy my name into char[] variable fname
        strcpy(fname, "Leah");

        // Create the linked list!
        first = (elementptr) malloc(sizeof(element));
        // (elementptr) is a type cast to change to pointer
        // malloc() is a function that allocates the memory
        // sizeof() is a function that allocates enough bytes to store element
        last = first;
        last -> letter = 'G';
        last -> data = 30;
        last -> next = NULL;

        // Use for loop to create rest of linked list and fill it in with random information
        for (i=0; i < strlen(fname); i++)
        {
                last -> next = (elementptr) malloc(sizeof(element));
                last = last -> next;
                last -> letter = fname[i];
                last -> data = i;
                last -> next = NULL;
        }

        // See what the linked list looks like
        printf("\nHere's what the linked list looks like now: \n");

        trav_print(first);


        // Example of deleting an element from anywhere in a linked list
        printf("\nNow let's see an example of deleting an element from the linked list: \n");

        current = first; // Adjust this as needed for learning
        printf("The current pointer is currently at the element with letter %c and data %d.\n",
current -> letter, current -> data);

        if (current == first)
        {
                first = first -> next;
                free(current);
        }
        else
        {
                temp = first;

                while(temp -> next != current)
                {
                        temp = temp -> next;
                }

                temp -> next = current -> next;

                free(current);

                if (temp -> next == NULL)
                {
                        last = temp;
                }
        }
        trav_print(first);

        printf("\n***\n\nThis is the end of the program.\n\n***\n");

        free(first);
        free(temp);
        free(last);

        return 0;
}

// Function definition for the trav_print function
void trav_print(elementptr f)
{
        // f signifies the first pointer
        elementptr c; // initialize pointer c for current

        c = f;

        if (c == NULL)
                printf("There is no linked list!\n");
        else
                while (c != NULL)
                {
                        printf("The letter is currently %c\n", c -> letter);
                        printf("The data is currently %d\n", c -> data);
                        c = c -> next;
                }
        printf("\n");
```

Example of deleting from LL



Program Output:

```
Let's see an example of linked lists in action!
Ready?

***

This is the beginning of the program.

***

Here's what the linked list looks like now:
The letter is currently G
The data is currently 30
The letter is currently L
The data is currently 0
The letter is currently e
The data is currently 1
The letter is currently a
The data is currently 2
The letter is currently h
The data is currently 3

Now let's see an example of deleting an element from the linked list:
The current pointer is currently at the element with letter G and data 30.
The letter is currently L
The data is currently 0

The letter is currently e
The data is currently 1
The letter is currently a
The data is currently 2
The letter is currently h
The data is currently 3


***

This is the end of the program.

***
```

what LL looks like initially

what LL looks like now