Creating a linked list-last time, I created three nodes then linked them up. Here I am creating nodes with a loop:

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

struct node
{
  int number;
  struct node *next;
};

/*create a node. When head is NULL (meaning we're creating the first node), notice that temp->next is equal to NULL (meaning the first node is created)*/
struct node* addNode(struct node* head, int n)
{
  struct node *temp = malloc( sizeof( struct node) );
  temp->number = n;
  temp->next = head;
  return temp;
}
int main(void)
{
  struct node *head = NULL, *temp;
  int i;

  for(i = 0; i < 3; i++)
  {
    head = addNode(head, i);
  }

  while(head != NULL)
  {
    printf("freeing: %d\n", head->number);
    temp = head->next;
    free( head );
    head = temp;
  }
}
```

Program 1: ABC Call Center handles people waiting to speak with a representative (information held in a file) by having each representative handle calls from a different city. Create a program for employees that allows them to

type in the city they are handling calls from. The program should also keep track of how many total minutes are spent handling calls.

```
[fiq8745@omega ~]$ gcc -o callcenter call.c
[fiq8745@omega ~]$ ./callcenter

Call center loading...session started.
Which city are we taking callers from?
Fort Worth

--Now on the line: Nick
Minutes: 1

Minutes: 2

Minutes: 3
d
Call completed!

--Now on the line: Baer
Minutes: 1

Minutes: 2

Minutes: 3

Minutes: 4

Minutes: 5
d
Call completed!

--Now on the line: Julio
Minutes: 1

Minutes: 2

Minutes: 3
d
Call completed!

Total time (in minutes) spent for callers from Fort Worth: 11
Ending session...
```

Code:

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

```c
#include <string.h>

typedef struct Node
{
    char *name;
    char* phone_num;
    char *city;
    struct Node *next;
}Node;

Node* populate_list()
{
   FILE *fp=fopen("callcenter.txt", "r");

   char *token;
   char *line=malloc(100);
   struct Node *head = NULL;
   struct Node *temp=NULL;

    while(fgets(line, 100,fp))
    {
      token=strtok(line, ",");
      temp = malloc(sizeof(Node));
      temp->name=malloc(20);
      strcpy(temp->name,token);
      token=strtok(NULL, ",");
      temp->phone_num=malloc(20);
      strcpy(temp->phone_num, token);

      temp->city=malloc(20);
      if(atoi(token)==817)  /*will go up to the -, get the first 3 nums */
      {
          strcpy(temp->city,"Fort Worth");
      }

      else if(atoi(token)==214)
      {
          strcpy(temp->city,"Dallas");
      }

      else
      {
          strcpy(temp->city,"Grand Prairie");
      }

      temp->next=head;
      head=temp;

    //How the items will look out:
    //printf("pPhone number:%s, city: %s, %d\n", temp->phone_num,temp->city, atoi(token));
    //Phone number:817-966-7771
    //, city: Fort Worth, 817
    }

    free (line);
    return head;
```

```
}

Node* pick_next(Node*list, char*city_name)
{
  while(list)
  {
    if(!strcmp(list->city, city_name))
    {
      return list;
    }
     list=list->next;
  }
  return NULL; /*we can't find anyone from that city*/
}

Node *completed(Node *current, char *name)
{
  Node* temp=NULL;
  Node* head=current;

  if(!current) /*handle empty list, current==NULL */
  {
    return current;
  }

  if(!strcmp(current->name, name)) /*handle case that node to delete is the first one in the list*/
  {
      printf("Call completed!\n");
      temp=current; /*save original head of list...to delete*/
      current=current->next; /*new head of list is the next one*/
      free(temp);
      return current;
  }

  current=current->next; //set to second node (since we know first is not the name)

  while(current)
  {
    if(!strcmp(name, current->name)) /*found node to delete*/
    {
      temp->next=current->next;
      free(current);
      printf("Call completed!\n");
      break;
    }
      temp=current; //hold on to previous during next round
      current=current->next;
  }
  return head;
}

void show_list(Node* head)
{
  printf("\n--Current list--\n");
  while(head)
  {
    printf("%s  \n", head->name);
    head=head->next;
  }
}


int main(int argc, char **argv)
{
  Node* call_wait=populate_list();
  char *answer=malloc(20);
  Node* cur=call_wait;
  char c='a';
  int minutes=0, total=0; /*starting minutes, total, loop*/

  printf("\nCall center loading...session started.");
  show_list(call_wait);
  printf("\nWhich city are we taking callers from?\n");
  fgets(answer,20,stdin);
  strtok(answer,"\n");

  while(cur)
  {
      cur=pick_next(call_wait, answer);
      if(cur) /*no more callers from this city*/
      {
        printf("\n--Now on the line: %s\n", cur->name);

        while(c!='d')
        {
          minutes++;
          printf("Minutes: %d\n", minutes);
          scanf(" %c", &c);
        }

        call_wait=completed(call_wait, cur->name); /*you could have deleted node directly in the
pick_next function*/
        c='a';
        total++;
      }
  }

  printf("\nTotal time (in minutes) spent for callers from %s: %d\n", answer,total);
  show_list(call_wait);
  printf("Ending session...\n");
}
```

Program 2: ABC Realty has asked you to create a program that allows the user to update the inventory in real time (adding houses). All new houses should be

added to the beginning of the list. In addition, a user should be able to search for a house with a given budget.

```
[fiq8745@omega ~]$ gcc -o house house.c
[fiq8745@omega ~]$ ./house housestuff.txt

***Welcome to ABC Realty.***
Update inventory or find house?
find
What is your budget? $700000

Price: $350000    1212 Geo Street
Price: $500500    1212 Londe Drive
Price: $400000    1212 Cherry Lane
Price: $130000    3762 Ashley Ct
--Houses that match your budget: 4

***Welcome to ABC Realty.***
Update inventory or find house?
update

***Adding a new house:***
Enter address: 1234 New House
Enter city: Dallas
Enter price: $450000
New house added!


***Welcome to ABC Realty.***
Update inventory or find house?
find
What is your budget? $700000

Price: $450000    1234 New House
Price: $350000    1212 Geo Street
Price: $500500    1212 Londe Drive
Price: $400000    1212 Cherry Lane
Price: $130000    3762 Ashley Ct

--Houses that match your budget: 5

***Welcome to ABC Realty.***
Update inventory or find house?
quit
Exiting...
```

Code: (I STRONGLY ADVISE YOU TO DRAW THE POINTERS OUT ON PAPER LIKE I DO IN CLASS)

```
/*house listings*/
#include <stdio.h>
```

```c
#include <stdlib.h>
#include<string.h>

struct node
{
    char *address;
    char *city;
    int price;
    struct node *next;
};

/*returning the head (address) of the linked list*/
struct node* populate_list()
{
    FILE *fp=fopen("housestuff.txt", "r"); /*can pass filename as an argument*/

    char *token;
    char *line=malloc(100);
    struct node *head = NULL;
    struct node *temp=NULL;

    int i=0;

    while(fgets(line, 100,fp))
    {
        token=strtok(line, ",");

        temp = malloc(sizeof(struct node));
        temp->address=malloc(20);
        strcpy(temp->address,token);

        token=strtok(NULL, ",");
        temp->city=malloc(20);
        strcpy(temp->city, token);

        token=strtok(NULL,",\n");
        temp->price=atoi(token);

        temp->next=head;
        head=temp;
    }

    return head;
}


struct node* add_house(struct node* h) /*adds at beginning of list-insert*/
{
  char *answer=malloc(20);
  printf("\n***Adding a new house:***\n");
  struct node *new_house=malloc(sizeof(struct node));

  new_house->address=malloc(20);
  printf("Enter address: ");
  fgets(answer,20,stdin);
  strtok(answer, "\n");
  strcpy(new_house->address, answer);

  new_house->city=malloc(20);
  printf("Enter city: ");
  fgets(answer, 20, stdin);
  strtok(answer,"\n");
  strcpy(new_house->city, answer);

  printf("Enter price: $");
  fgets(answer, 20, stdin);
  strtok(answer,"\n");
  new_house->price=atoi(answer);

  new_house->next=h;  /*the added house becomes new head of the linked list*/

  printf("New house added!\n\n");

  return new_house; /*return the new head*/

}

void print_options(struct node *h, int total)
{
  int i=0;
  printf("\n");

  while(h!=NULL)
  {
    if(h->price<=total)
    {
      printf("Price: $%d    %s \n",h->price, h->address);
      i++;
    }
    h=h->next;

  }

  printf("--Houses that match your budget: %d\n", i);

}


int main(int argc,char **argv)
{
  struct node *houses=populate_list();
  char *answer=malloc(20);
  int total=0;


  while(answer)
  {
    printf("\n***Welcome to ABC Realty.***\nUpdate inventory or find house?\n");
    fgets(answer, 20, stdin);
    strtok(answer,"\n");
    if(!strcmp(answer, "update"))
    {
      houses=add_house(houses);
    }

    else if(!strcmp(answer,"find"))
    {
      printf("What is your budget? $");
      fgets(answer, 20, stdin);
      strtok(answer,"\n");
      total=atoi(answer);
      print_options(houses, total);
    }

    else if(!strcmp(answer,"quit"))
    {
      answer=NULL;
      printf("Exiting...\n");
    }

    else
    {
      printf("Invalid entry.\n");
    }
  }

  //dont forget to free

}
```

Extra: Concatenate linked lists

```
computer$ ./a.out

--Enter student name: Frank /*make first list*/
Enter grade: 89

--Enter student name: Jon
Enter grade: 90

--Enter student name: Jane /*make second list*/
Enter grade: 100

--Enter student name: Bob
Enter grade: 30

Info as entered:
Jon  Frank  Bob  Jane /*printing out single concatenated list*/
```

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

```
typedef struct student_grades{

  char *name;
  int grade;
  struct student_grades* next;

}Student;


Student* enter_info(int num_students)
{
  char* answer=malloc(50);
  /*should check if answer is null to continue*/
  Student* head=NULL;
  Student* temp=NULL;
  int i;

  for(i=0;i<num_students;i++)
  {
    temp=malloc(sizeof(Student)); /*should check if null*/
    printf("\n--Enter student name: ");
    fgets(answer,50,stdin);
    strtok(answer,"\n");

    temp->name=malloc(50);
    strcpy(temp->name,answer);

    printf("Enter grade: ");
    fgets(answer,50,stdin);
    strtok(answer,"\n");
    temp->grade=atoi(answer);

    temp->next=head;
    head=temp;
  }

  return head;
}

Student* concatenate_lists(Student* all_students1, Student* all_students2)
{
    Student*previous=NULL;
    Student*head=all_students1;

    while (all_students1) /*get to end of first list… while all_students1 != NULL */
    {
        previous=all_students1;
        all_students1 = all_students1->next;
    }

    previous->next=all_students2; /*once we are at end, we add beginning of second list to end of first list*/
    return head;
}

void print_list(Student* head, char *message)
{
    Student* temp = head;
    printf("\n%s\n", message);

    while (temp != NULL)
    {
        printf("%s  ", temp->name);
        temp = temp->next;
    }

    printf("\n");
}


int main(int argc, char** argv)
{
  Student* all_students=enter_info(2); /*create two linked lists*/
  Student* all_students2=enter_info(2);

  all_students=concatenate_lists(all_students, all_students2);
  print_list(all_students, "Info as entered: ");

}
```

Reversing linked lists:

```
computer$ ./a.out

--Enter student name: Bill
Enter grade: 99

--Enter student name: Bob
Enter grade: 100

--Enter student name: Benny
Enter grade: 78

Info as entered:
Benny  Bob  Bill

Reverse:
Bill  Bob  Benny /*list is now reversed*/
```

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

```
typedef struct student_grades{

  char *name;
  int grade;
  struct student_grades* next;

}Student;


Student* enter_info(int num_students)
{
  char* answer=malloc(50);
  Student* head=NULL;
  Student* temp=NULL;
  int i;

  for(i=0;i<num_students;i++)
  {
    temp=malloc(sizeof(Student));
    printf("\n--Enter student name: ");
    fgets(answer,50,stdin);
    strtok(answer,"\n");

    temp->name=malloc(50);
    strcpy(temp->name,answer);

    printf("Enter grade: ");
    fgets(answer,50,stdin);
    strtok(answer,"\n");
    temp->grade=atoi(answer);

    temp->next=head;
    head=temp;
  }

  return head;
}

Student* reverse(Student* head_ref)
{
    Student* prev = NULL;
    Student* current = head_ref;
    Student* next = NULL;

    while (current) { /*while  current != NULL*/

        next = current->next;

        current->next = prev;
        prev = current;
        current = next;
    }

    return prev;
}



void print_list(Student* head, char *message)
{
    Student* temp = head;
    printf("\n%s\n", message);

    while (temp != NULL)
    {
        printf("%s  ", temp->name);
        temp = temp->next;
    }

    printf("\n");
}



int main(int argc, char** argv)
{
  Student* all_students=enter_info(3);
  print_list(all_students, "Info as entered: ");

  Student* reversed=reverse(all_students);
  print_list(reversed, "Reverse: ");

  /*free*/

}
```