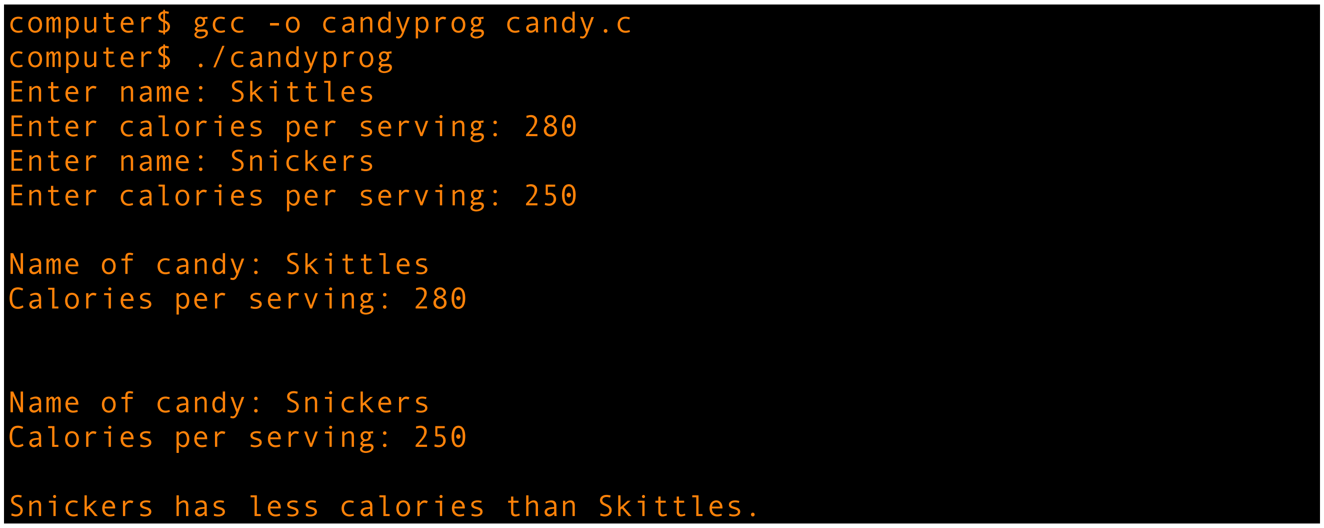
```
computer$ gcc lottostuff.c
computer$ ./a.out lotto.txt

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Sunday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 7 13 21 42 2
3

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Monday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 4 22 1 35 12
23

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Tuesday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 20 3 11 12 1
16

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Wednesday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 19 1 28 13
42 41

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Thursday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 21 42 2 3 7
13

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Friday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 35 12 23 4
22 1

See today's lotto numbers? Enter 1 for yes, 2 for no: 1

--Today is: Saturday!!!!
For your chance to win **5 billion dollars**, lotto numbers are: 11 12 20 3 1
16

Exiting program...
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define ALL_DAYS 7

int read_file(int val[][6], char *filename)
```

```c
{
  FILE *fp;
  char *mode="r";
  int status;
  fp=fopen(filename, mode);

  if(!fp)
  {
    printf("No file found.\n");
    status=0;
  }
/*note that since I'm not actually using lotto numbers directly, just printing, I could have held
each line of numbers and a string and then printed that out.  I just wanted you guys to get
practice with breaking down values from a file and storing them in an array*/
  else
  {
    status=1;
    int i=0,j=0;

    int current, previous=0;
    char line[100];
    char *token;

    while (!feof(fp))
    {
      fgets(line,100,fp); /*gets whole line from the file of lotto nums: "6,4,3,2,2,1"*/
      token=strtok(line, ","); /*the first comma becomes \0 and token points at the 6 in
"6\04,3,2,2,1"*/

      while(token) /*token will only equal null when there are no more items on the line)...can
also write while(token!=NULL)*/
      {
        val[j][i]=atoi(token); /*turn the first token into an int and place in the array*/
        token=strtok(NULL, ",\n");
        i++;
      }
      i=0; //reset i for next line in the file
      j++;
    }
    fclose(fp);
  }

  return status;
}

void run_lotto(char *day, int todays_nums[])
{
  int i;
  printf("\n--Today is: %s!!!!\n", day);
  printf("For your chance to win **5 billion dollars**, lotto numbers are: ");

  for(i=0;i<6;i++)
  {
    printf("%d ", todays_nums[i]);
  }

  printf("\n");

}


int main(int argc, char **argv)
{
  int lotto_nums[ALL_DAYS][6];
  int status=0, counter=0;
  char* days[]={"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};

  char **days_ptr=days;
  int (*lotto_ptr)[6]=lotto_nums;


  status=read_file(lotto_nums, argv[1]);

  if(status)/*file successfully read*/
  {

    while(counter<ALL_DAYS)
    {
      printf("\nSee today's lotto numbers? Enter 1 for yes, 2 for no: ");
      scanf("%d", &status);/*don't need the status variable anymore so reusing it*/

      if(status==1)/*yes*/
      {
        run_lotto(*days_ptr, *lotto_ptr);
        days_ptr++;
        lotto_ptr++;
        counter++;
      }

      else/*assume entered 2 for no*/
      {
        counter=10; /*just picked a random larger number than 7 to break out of loop*/
      }
    }
  }

  printf("\nExiting program...\n");

}
```

Program 1



```
computer$ gcc -o candyprog candy.c
computer$ ./candyprog
Enter name: Skittles
Enter calories per serving: 280
Enter name: Snickers
Enter calories per serving: 250

Name of candy: Skittles
Calories per serving: 280


Name of candy: Snickers
Calories per serving: 250

Snickers has less calories than Skittles.
```

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

struct Candy{ /*defining the struct*/

        char name[20];
        int calories;

};
typedef struct Candy C; /*using typedef so now we can just call it C (instead of struct Candy)*/

/*this function prints out candy information*/
void print_info(C c1)
{
        printf("Name of candy: %s\n", c1.name);
        printf("Calories per serving: %d\n",c1.calories);

        /*c1.calories=400; if we put something like this it would not work outside the function because
we are passing in a single struct by value (meaning any change would be on a copy of the struct) and
not by reference*/
}


/*this function compares two candies and returns a value based on which has more calories*/
int compare_candy(C c1, C c2)
{
        int ret;

        if(c1.calories<c2.calories)
```

```
	{
		ret=1;
	}

	else if(c2.calories<c1.calories)
	{
		ret=-1;
	}

	else
	{
		ret=0;
	}

	return ret;
}

int main(int argc, char**argv)
{
	int i=0;
	char line[40];
	C candy1[2]; /*since we used typedef, we can use our struct just like we use int for example*/

	while(i<2)
	{
		printf("Enter name: ");
		scanf("%s", line);
		strcpy(candy1[i].name,line);
		printf("Enter calories per serving: ");
		scanf("%s", line);
		candy1[i].calories=atoi(line);
		i++;
	}

	print_info(candy1[0]);
	print_info(candy1[1]);

	i=compare_candy(candy1[0], candy1[1]);

	if(i==1 || i==-1) /*you can also just say i since 1 and -1 are true*/
	{
		if(i==1) /*you need to specify i==1 here since we are distinguishing between 1 and -1*/
		{
			printf("%s has less calories than %s.\n", candy1[0].name, candy1[1].name);
		}

		else
		{
			printf("%s has less calories than %s.\n", candy1[1].name, candy1[0].name);
		}
```

```
        }

        else
        {
                printf("They're the same.\n");
        }
}
```

Program 2

```
computer$ gcc -o movie movie.c
computer$ ./movie
Enter movie name: movie1
Enter rating: 3
Enter movie name: movie2
Enter rating: 4

***Movie info***
Name of movie: movie1
Rating of movie: 3

***Movie info***
Name of movie: movie2
Rating of movie: 4

---Changing both ratings to 1:

***Movie info***
Name of movie: movie1
Rating of movie: 1

***Movie info***
Name of movie: movie2
Rating of movie: 1
```

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

struct movie{

        char name[20];
        int rating;

}; /*not using typedef here*/

/*since we're using a pointer, we can input values into our structs in function.  We couldn't do this in
the previous program since we were passing in a copy of the struct (by value)*/
void enter_info(struct movie *m)
```

```
{
	char line[20];
	printf("Enter movie name: ");
	scanf("%s", line);
	strcpy(m->name, line); /*notice we are accessing the member using ->.  This is used since are
using a pointer.  If we are not using a pointer, we use the normal . to access members of our struct.
Note that  -> is shorthand for *m.name (deref with * then access member with .)    */

	printf("Enter rating: ");
	scanf("%s", line);
	m->rating=atoi(line);
}

/*this function prints out movie info*/
void print_info(struct movie *m)
{
	printf("\n***Movie info***\n");
	printf("Name of movie: %s\n", m->name);
	printf("Rating of movie: %d\n", m->rating);
}


int main(int argc, char**argv)
{
	struct movie m1; /*since I didn't use typedef, I have to declare using struct*/
	struct movie m2;

	struct movie *m22=&m1; /*you can create a pointer at a struct just like at int or char*/

	enter_info(m22);
	enter_info(&m2);

	print_info(m22);
	print_info(&m2);

	printf("\n---Changing both ratings to 1:\n");
	m1.rating=1; /*change the ratings*/
	m2.rating=1;

	print_info(&m1); /*print out changed ratings*/
	print_info(&m2);


}
```