Program 1:

```
computer$ gcc fortune.c
computer$ ./a.out <cookie.txt //opening a file using redirection
message: Good things will happen to you
Lucky nums are:
34
11
3
5
6
4
Computers-MacBook-Air:C computer$ ./a.out<cookie.txt>output.txt //now
it goes to a file (named output.txt) instead of screen
```

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

typedef struct cookie{
	char message[100];
	int lucky_nums[6];
}fortune_cookie;


void fill_cookies(fortune_cookie*f)
{
	char line[500];
	int i,j;
	char *token;


	fgets(line, 500, stdin);
	token=strtok(line, ",");

	for(i=0;i<2;i++)
	{
		token=strtok(NULL, ",");
		strcpy(f[i].message,token);
		for(j=0;j<6;j++)
		{
			token=strtok(NULL, ",\n");
			f[i].lucky_nums[j]=atoi(token);
		}
	}
}

void cookie_info(fortune_cookie*f, int num)
```

```
{
	printf("message: %s\n",f[num].message);
	int i;
	printf("Lucky nums are:\n");
	for(i=0;i<6;i++)
	{
		printf("%d\n", f[num].lucky_nums[i]);

	}
}

int main (int argc, char **argv)
{
	fortune_cookie f1[2];
	fill_cookies(f1);
	cookie_info(f1, 1);
}
```

Program 2:

```
computer$ gcc -o irish irish.c
computer$ ./irish
Enter the file name:
irishnames1.txt
Info successfully read from file.
Enter name to look for and the language it is in: Daithi Irish
Found a match!
The English equivalent for Daithi is David.
computer$ ./irish
Enter the file name:
irishnames1.txt
Info successfully read from file.
Enter name to look for and the language it is in: Dylan English
Found a match!
The Irish equivalent for Dylan is Dillon.
```

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

typedef struct irish{

	char irish_name[20];
	char pick;
	char english_equiv[20];


}irish;
```

```
int read_file(int nums, irish names[]);
void find_name(char* name, char *lang, int size, irish *names); /*Note-this function doesn't catch if a person doesn't enter an actual name*/

int main (int argc, char **argv) {

	irish names[13];
	char n[10];
	char lang[10];
	int s=sizeof(names)/sizeof(names[0]);
	int result=read_file(s, names);

 if(result)/*file opened*/
 {
	printf("Enter name to look for and the language it is in: ");
	scanf("%s %s", n, lang);

	find_name(n, lang, s, names);
 }

  return 0;
}

/*returns 0 if file didn't open, 1 if it did*/
int read_file(int nums, irish names[])
{
	char filename[20];
	printf("Enter the file name:\n");
	scanf("%s", filename);

	FILE *fp=fopen(filename, "r+");

	if(fp!=NULL)
	{
		char line[50];
		int i=0;
		char *token;

		while(i<nums)
		{
			fgets(line, 50, fp);
			token=strtok(line, ",");
			strcpy(names[i].irish_name, token);
			token=strtok(NULL, ",\n");
			names[i].pick=token[0];
			token=strtok(NULL, ",\n");
			strcpy(names[i].english_equiv, token);
			i++;
```

```
                }

                printf("Info successfully read from file.\n");
                 fclose(fp);
                return 1;
        }
        else
        {
                printf("File didnt open. Exiting...\n");
                return 0;
        }

}

/*notice that I am using different ways to access the information (-> and * .) */
void find_name(char* name, char *lang, int size, irish *names)
{
        int i;
        int index;

        if(!strcmp(lang, "Irish")) /*irish name given*/
        {
                for(i=0;i<size;i++)
                {
                        if(!strcmp((*names).irish_name, name))
                        {
                                printf("Found a match!\n");
                                printf("The English equivalent for %s is %s.\n", name, names->english_equiv);  /*using ->*/
                                break;
                        }
                        names++;
                }
        }

        else /*english name given*/
        {
                for(i=0;i<size;i++)
                {
                        if(!strcmp(names->english_equiv, name))
                        {
                                printf("Found a match!\n");
                                printf("The Irish equivalent for %s is %s.\n", name, (*names).irish_name); /*using * (deref)
and . (dot) */
                                break;
                        }

                        names++;
                }
        }
}
```