```c
/* lab1.c
 * Christopher Brant
 * cbrant
 * ECE 2230
 * Section 001
 * Spring 2017
 * Programming Assignment #1
 * Due on 1/23/17 at 11:30 PM
 * Professor Walt Ligon
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "inventory.h"

#define MAXLINE 50
#define MAXCOMSIZE 5
#define MAXDESCRIPTION 15

int main(int argc, char *argv[])
{
        char line[MAXLINE];                                             // Input buffer
        char command[MAXCOMSIZE];                                       // Command string
        char item_description[MAXDESCRIPTION];          // Description string
        char overflow[MAXDESCRIPTION];                          // Overflow input validation check variable
        float temp_power;                                               // Input variable for power
        int item_info;                                                  // Input variable for item info
        int keylook, keynum = 0;                                        // Item key lookup number and item key number counter
        int cont = 1;                                                   // Continue variable
        int reterr;                                                     // The return error variable
        struct inventory_item *slotnum, *newitem;       // New item and searched item pointers

        // The following variable is the inventory pointer
        struct inventory *inv1 = inventory_create();

        /* Continue to wait for input and execute commands until
           the QUIT command is entered */
        while (cont == 1)
        {
                overflow[0] = '\0';
                printf("\nCommands:\n> ADD\n> LOOK key-data\n> DEL key-data\n> LIST\n> QUIT\n\n");

                reterr = 0;
                fgets(line, sizeof(line), stdin);       // Gather user input from the keyboard
                sscanf(line, "%s %d %s", command, &keylook, overflow);

                // Conditions for the ADD command and follow up questions
                if (strcmp(command, "ADD") == 0 && strlen(line+1) == 3 && overflow[0] == '\0')
                {
                        newitem = (struct inventory_item *)malloc(sizeof(struct inventory_item));
                        newitem->item_key = keynum++;

                        printf("\nWhat is the item type: ");     // The following four blocks of code are
                        fgets(line, sizeof(line), stdin);               // used to collect inventory item info
                        sscanf(line, "%d", &item_info);                 // from the user.
                        newitem->item_type = item_info;

                        printf("\nWhat is the item description:");
                        fgets(item_description, sizeof(item_description), stdin);
                        sscanf(item_description, "%s", newitem->description);

                        printf("\nWhat is the item power: ");
                        fgets(line, sizeof(line), stdin);
                        sscanf(line, "%f", &temp_power);
                        newitem->power = temp_power;

                        printf("\nWhat is the item modifier: ");
                        fgets(line, sizeof(line), stdin);
                        sscanf(line, "%d", &item_info);
                        newitem->modifier = item_info;


                        reterr = inventory_add(inv1, newitem);

                        if (reterr == 0)
                                printf("Data added. Assigned item key is %d\n", keynum-1);
                        else
                                printf("Error. Item not added or inventory is full.\n");
                }

                // Conditions for the LOOK command and follow up print statements
                else if (strcmp(command, "LOOK") == 0 && overflow[0] == '\0')
                {
                        printf("Looking for inventory item with item key: %d\n", keylook);

                        slotnum = inventory_lookup(inv1, keylook);

                        if (slotnum == NULL)
                                printf("Data not found with item key: %d\n", keylook);
                        else
                        {
                                printf("\nData found\n");
                                printf(" Key:%d", slotnum->item_key);
                                printf(" Type:%d", slotnum->item_type);
                                printf(" Description:%s", slotnum->description);
                                printf(" Power:%f", slotnum->power);
                                printf(" Modifier:%d\n", slotnum->modifier);
                        }
                }

                // Conditions for the DEL command and follow up print statements
                else if (strcmp(command, "DEL") == 0 && overflow[0] == '\0')
                {
                        reterr = inventory_delete(inv1, keylook);

                        if (reterr == 0)
                                printf("Data deleted.\n");
                        else
                                printf("Data not found.\n");
                }

                // Conditions for the LIST command and continual print statements
                else if (strcmp(command, "LIST") == 0 && strlen(line+1) == 4 && ove
```

```
rflow[0] == '\0')
                {
                        slotnum = inventory_first(inv1);

                        if (slotnum == NULL)
                                printf("Inventory empty.\n");

                        while (slotnum != NULL)
                        {
                                printf("\nKey:%d", slotnum->item_key);
                                printf(" Type:%d", slotnum->item_type);
                                printf(" Description:%s", slotnum->description);
                                printf(" Power:%f", slotnum->power);
                                printf(" Modifier:%d\n", slotnum->modifier);
                                slotnum = inventory_next(inv1);
                        }
                }

                // Conditions for the QUIT command and the possible errors
                else if (strcmp(command, "QUIT") == 0 && strlen(line+1) == 4 && ove
rflow[0] == '\0')
                {
                        cont = 0;
                        reterr = inventory_destroy(inv1);

                        if (reterr == 0)
                                printf("Inventory destroyed. Quitting program.\n");
                        else
                                printf("Error. Could not destroy inventory. Quittin
g program.\n");
                }

                // Conditions for incorrect inputs
                else
                {
                        printf("Incorrect command. Try again or type 'QUIT' to end
all processes.\n");
                        printf("Remember no spaces after single word commands.\n");
                }

        }

        return 0;
}
```