```c
// Name: Rodrigo Ignacio Rojas Garcia
// Course Number: ECE 2230
// Section: 001
// Semester: Spring 2017
// Assignment Number: 1

// Library Declaration Section
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "inventory.h"
#include <ctype.h>

int main(int argc, char *argv[])
{
    // Variable Declaration Section
    char input[MAXCHARACTERS]; // Used to store commands temporarily
    char input2[MAXCHARACTERS]; // Used specifically to store description command o
f items temporarily
    char command[MAXCHARACTERS]; // Used to store command entered by user
    char description[MAXDESCRIPTION]; // Ised specifially to store description ente
red by user
    int c1 = 0; // Used in loops as counter
    int key_data = -17; // Used to store key_data command
    int addition_result = -1; // Used to verify if inventory_add was successful
    int delete_result = -1; // Used to verify that iventory_delete was successful
    int destroy_result = -1; // Used to verify that inventory_destroy was successfu
l
    char error = -17; // Used as a dummy variable to see if command entered had mor
e than arguments needed
    struct inventory *inventory_pointer;

    // Function iventory_create() allocates memory for structure pointer inventory_
pointer which points to a creted inventory.
    inventory_pointer = inventory_create();

    // Sets the first_item to -1 which symbolizes that the first_item hasn't been f
ound yet
    inventory_pointer->first_item = -1;

    // For loop will fill integer array free_slots with integer -1 to signify that
the sloot is able store items desired
    for (c1 = 0; c1 < MAXITEMS; c1++)
    {
        inventory_pointer->allocated_slots[c1] = -1;
    }

    // While loop allows program to run indifinitely until user enters command QUIT
    while (1)
    {

        key_data = -17;
        error = -17;
        // Prints the Menu Code for the user and asks user to enter a command
        printf("Please enter any of the following commands (key_data must be an int
eger greater or equal to 0):\n\n");
        printf("ADD key_data\nLOOK key_data\nDEL key_data\nLIST\nQUIT\n\n");
        printf("Command: ");

         // Allows the user to enter a command from the Menu and stores it into var
iable command
        fgets(input, MAXCHARACTERS, stdin);
        sscanf(input, "%s %d %s", command,&key_data, &error);
        // Insers the NULL character into the character array variable Command
        c1 = c1 + 1;

        printf("\n");

        // If statement will be accessed if user enter the command ADD
        if (strcmp("ADD", command) == 0 && key_data >= 0 && error == -17)
        {

            // Creates structure of an item each time loop runs and allocates memor
y for it as well.
            struct inventory_item *item_pointer;
            item_pointer = (struct inventory_item*)calloc(1, sizeof(struct inventor
y_item));

            item_pointer->item_key = key_data;

            // Asks the user to enter the item type and stores it into item_pointer
 structure in variable item_type
            printf("Please enter the item type (0 -Armor, 1 - Sword, 2- Shield, 3 -
 Scroll, 4 - Potion): \n");
            printf("Item type: ");
            fgets(input, MAXCHARACTERS, stdin);
            sscanf(input, "%d", &item_pointer->item_type);
            printf("\n");

            // Asks the user to enter a short description of item entered and store
s it in structure item_pointer in variable description
            printf("Please enter the description of the item entered (15 characters
 maximum): \n");
            printf("Description: ");
            fgets(input2, MAXCHARACTERS, stdin);
            for (c1 = 0; c1 < MAXDESCRIPTION - 1; c1++)
            {
                description[c1] = input2[c1];
            }
            description[c1] = '\0';
            strcpy(item_pointer->description, description);
            printf("\n");

            // Asks the user to enter the power of item entered and stores it in st
ructure item_pointer in variable power
            printf("Please enter the power of the item entered (zero is useless, ne
gative has opposite effect): \n");
            printf("Power: ");
            fgets(input, MAXCHARACTERS, stdin);
            sscanf(input, "%f", &item_pointer->power);
            printf("\n");

            // Asks the user to enter the power of item entered and stores it in st
rucutre item_pointer in variable modifer
            printf("Please enter the modifier of the item entered: \n");
            printf("Modifier: ");
            fgets(input, MAXCHARACTERS, stdin);
            sscanf(input, "%d", &item_pointer->modifier);
            printf("\n");

            addition_result = inventory_add(inventory_pointer,item_pointer);

            // If function addition_result returns a 0, it means the the item slot
was empty and item was added
            if (addition_result == 0)
            {
                printf("Data added.\n\n");
            }
            // If the addition_result returns a -1, it means the slot was already f
ull.
            else
            {
                free(item_pointer);
                printf("Data could not be added\n\n");
            }
        }

        // If statement will be accessed if user enters command "LOOK"
        else if (strcmp("LOOK", command) == 0 && key_data >=  0 && error == -17)
        {
            // Creates an inventory_item structure pointer named item_pointer
            struct inventory_item *item_pointer;

            // Function inventory_lookup will return the address of the key_data if
 found, if not it will return a NULL
            item_pointer = inventory_lookup(inventory_pointer, key_data);

            // If key_data found, it will print all the data of the item
            if (item_pointer != NULL)
            {
                printf("Key data: %d\n", item_pointer->item_key);
                printf("Item Type: %d\n", item_pointer->item_type);
                printf("Description: %s", item_pointer->description);
                printf("Power: %.2f\n", item_pointer->power);
                printf("Modifier: %d\n\n", item_pointer->modifier);
            }
            // If key_data was not found, a NULL was returned therefore it means th
at Data was not found
            else
            {
                printf("Data not found\n\n");
            }
        }

        // If statement will be accessed if user enters command "DEL"
        else if (strcmp("DEL", command) == 0 && key_data >= 0 && error == -17)
        {

            // Function inventory_delete will return a 0 if the key_data was found
and dynamic memory was freed, if not found returns -1
            delete_result = inventory_delete(inventory_pointer, key_data);

            // If key_data was found, 0 was returned, therefore dynamic memory was
successfully allocated
            if (delete_result == 0)
            {
                printf("Data deleted\n\n");
            }
            // If key_data was not found, -1 was returned, therefore the data did n
ot exist
            else
            {
                printf("Data not found\n\n");
            }
        }

        // If statement will be accessed if user enters command "LIST"
        else if (strcmp("LIST", command) == 0 && key_data == -17)
        {
            // Creates an inventory_item structure pointer named item_pointer
            struct inventory_item *item_pointer;

            // Function inventory_first will return the address of the first slot w
ith allocated memory
            item_pointer = inventory_first(inventory_pointer);

            // If the function inventory_first returns an address to first slot, it
 prints it's information of the item
            if (item_pointer != NULL)
            {
                printf("Key data: %d\n", item_pointer->item_key);
                printf("Item Type: %d\n", item_pointer->item_type);
                printf("Description: %s", item_pointer->description);
                printf("Power: %.2f\n", item_pointer->power);
                printf("Modifier: %d\n\n", item_pointer->modifier);
                // For loop will go through each slot with allocated memory and pri
nt the information of the item
                for (c1 = 0; c1 < MAXITEMS; c1++)
                {
                    // Function inventory_next will return the address of allocated
 memory slot that is not the first found
                    item_pointer = inventory_next(inventory_pointer);
                    // If a slot was found with allocated memory, it will print the
 item's description
                    if (item_pointer != NULL)
                    {
                        printf("Key data: %d\n", item_pointer->item_key);
                        printf("Item Type: %d\n", item_pointer->item_type);
                        printf("Description: %s", item_pointer->description);
                        printf("Power: %.2f\n", item_pointer->power);
                        printf("Modifier: %d\n\n", item_pointer->modifier);
                    }
                }
            }
            // If there are no allocated items, it will return that there no data f
ound
            else
            {
                printf("Data not found\n\n");
            }

        }

        // If statement will be accessed if user enters command "QUIT" and will pro
ceed to terminate the program.
        else if (strcmp("QUIT", command) == 0 && key_data == -17)
        {
            // Funciton will free all allocated memory for the items before exiting
 the program
            destroy_result = inventory_destroy(inventory_pointer);

            // If all memory allocated for items have been freed, then it will free
 memory allocated for the inventory
            if (destroy_result == 0)
            {
                free(inventory_pointer);
            }
            // If there was an error freeing allocated memory, then it will display
 this error
            else
            {
                printf("Program could not free all memory\n\n");
            }

            // Informs the user that the program has been terminated by the QUIT co
mmand
            printf("Program terminated\n\n");
            // Exits the program
            exit(0);
        }

        // Error message printed if unexpected first word enterd or incorrect numbe
r of arguments is entered
        else
        {
            printf("Error: Command entered is an unexpected word or contains incorr
ect number of arguments.\n\n");
        }
```

```
    }
    return 0;
}
```