```c
// Name: Rodrigo Ignacio Rojas Garcia
// Course Number: ECE 2230
// Section: 001
// Semester: Spring 2017
// Assignment Number: 1.5

// Library Declaration Section
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "inventory.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 key_data; // Used to store key_data entered by the user
    int c1; // Used as a counter in loops
    int addition_result = -2;   // Used to check if funciton inventory_add was succ
esfull
    int delete_result = -2; // Used to check if function inventory_delete was succe
sfull
    int destroy_result = -2;    // Used to check if function inventory_destroy was
succesfull
    char error; // Used to see if user entered wrong arguments
    struct inventory *inventory_pointer;    // Used to create an inventory

    // Calls the function inventory_create which will allocate dynamic memory and r
eturn the address of allocated memory.
    inventory_pointer = inventory_create();

    // Prints the Menu Code for the user and asks user to enter a command
    printf("***************************************\n");
    printf("Type any of the following commands:\n");
    printf("> ADD item_key\n> LOOK item_key\n> DEL item_key\n> LIST\n> QUIT\n");
    printf("***************************************\n\n");

    // While loop allows program to run indifinitely until user enters command QUIT
    while (1)
    {
        key_data = -17;
        error = -17;

        printf("> ");

         // 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);

        // If statement will be accessed if user enter the command ADD
        if (strcmp("ADD", command) == 0 && key_data >= 0 && error == -17)
        {
            // Variable Declaration Section
            struct inventory_item *lookup_pointer;

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

            // If function inventory_lookup found an empty item slot this will run
            if (lookup_pointer == NULL)
```

```c
            {
                // Creates inventory_item structure pointer which will be allocated
 with memory to fill the item descriptions
                struct inventory_item *item_pointer;
                item_pointer = (struct inventory_item*)calloc(1, sizeof(struct inve
ntory_item));

                item_pointer->item_key = key_data;

                // Asks the user to enter the item type and stores it into item_poi
nter structure in variable item_type
                printf("Enter the following data:\n");
                printf("Item type (0-4): ");
                fgets(input, MAXCHARACTERS, stdin);
                sscanf(input, "%d", &item_pointer->item_type);


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


                // Asks the user to enter the power of item entered and stores it i
n structure item_pointer in variable power
                printf("Power (0 - Useless, Neg. Opposite Effect): ");
                fgets(input, MAXCHARACTERS, stdin);
                sscanf(input, "%f", &item_pointer->power);


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


                // addition_result calls function inventory_add which will see if d
ata can be added or not, if successfull will return 0, otherwise returns -1
                addition_result = inventory_add(inventory_pointer, item_pointer);

                // If the addition of item in inventory was successfull, prints mes
sage that data was added to the inventory
                if (addition_result == 0)
                {
                    printf("Data Added\n");
                }
                // If the addition of the item was unsuccessfull, frees memory not
needed and prints that data could not be added
                else
                {
                    free(item_pointer);
                    printf("Data could not be added.\n\n");
                }
            }
            // If an item had already been entered with key_data entered, it will d
isplay this error
            else
            {
                printf("Item with item_key %d already in inventory\n", key_data);
            }
```

```
        }

        // 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("\nKey data: %d\n", item_pointer->item_key);
                switch(item_pointer->item_type)
                {
                    case 0:
                        printf("Item Type: Unknown\n");
                        break;
                    case 1:
                        printf("Item Type: Potion\n");
                        break;
                    case 2:
                        printf("Item Type: Scroll\n");
                        break;
                    case 3:
                        printf("Item Type: Weapon\n");
                        break;
                    case 4:
                        printf("Item Type: Armor\n");
                        break;
                    default:
                        printf("Item Type: Error\n");
                        break;
                }
                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");
            }
        }

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

            // Inventory_item lookup_result is declared and set to obtain the addre
ss of desired item with specified key_data. If item found,
            // lookup_result will contain the address of item, if not it will be se
t to NULL
            struct inventory_item *lookup_result;
            lookup_result = inventory_lookup(inventory_pointer, key_data);

            // If key_data was found, 0 was returned, therefore dynamic memory was
successfully allocated
            if (lookup_result != NULL)
            {
                // Function inventory_delete will return a 0 if the key_data was fo
und and dynamic memory was freed, if not found returns -1
                delete_result = inventory_delete(inventory_pointer, key_data);

                // If the the fuction inventory_delete was succesfull, it returns 0
, this if statement runs, and frees the dynamic memory allocated for item
                if (delete_result == 0)
                {
                    free(lookup_result);
                    printf("Data deleted\n");
                }
                // If the function inventory_dete was not succesfull, it returns a
-1, which means that dynamic memory could not be freed
                else
                {
                    printf("Data could not be freed.\n\n");
                }
            }
            // If key_data was not found, -1 was returned, therefore the data did n
ot exist
            else
            {
                printf("Data not found\n");
            }

        }

        // If statement will be accessed if user enters command "LIST"
        else if (strcmp("LIST", command) == 0 && key_data == -17)
        {
            // Strucutre of type intenvory_item created, and then set to function i
nventory_first that will return the address of the first item that has been allocat
ed with
            // dynamic memory. If none found, it will return a NULL
            struct inventory_item *item_pointer;
            item_pointer = inventory_first(inventory_pointer);

            // If an item was located in the inventory, it will print it’s informat
ion and will continue to run a loop to locate the next item located in the inventor
y
            if (item_pointer != NULL)
            {
                printf("\nKey data: %d\n", item_pointer->item_key);
                switch(item_pointer->item_type)
                {
                    case 0:
                        printf("Item Type: Unknown\n");
                        break;
                    case 1:
                        printf("Item Type: Potion\n");
                        break;
                    case 2:
                        printf("Item Type: Scroll\n");
                        break;
                    case 3:
                        printf("Item Type: Weapon\n");
                        break;
                    case 4:
                        printf("Item Type: Armor\n");
                        break;
                    default:
                        printf("Item Type: Error\n");
                        break;
                }
                printf("Description: %s", item_pointer->description);
                printf("Power: %.2f\n", item_pointer->power);
                printf("Modifier: %d\n\n", item_pointer->modifier);

                // For loop will run unti it reaches the same number as MAXITEMS wh
ile trying to locate the next item in the inventory that has been allocated with me
mory
                for (c1 = inventory_pointer->location; c1 < MAXITEMS; c1++)
                {
                    // Function will return the address of the item in inventory if
 found, if not it will return a NULL
                    item_pointer = inventory_next(inventory_pointer);

                    // If function inventory_next return the address of the next it
em in inventory, it will print it’s information and continue to look for the next o
ne
                    if (item_pointer != NULL)
                    {
                        printf("Key data: %d\n", item_pointer->item_key);
                        switch(item_pointer->item_type)
                        {
                            case 0:
                                printf("Item Type: Unknown\n");
                                break;
                            case 1:
                                printf("Item Type: Potion\n");
                                break;
                            case 2:
                                printf("Item Type: Scroll\n");
                                break;
                            case 3:
                                printf("Item Type: Weapon\n");
                                break;
                            case 4:
                                printf("Item Type: Armor\n");
                                break;
                            default:
                                printf("Item Type: Error\n");
                                break;
                         }
                        printf("Description: %s", item_pointer->description);
                        printf("Power: %.2f\n", item_pointer->power);
                        printf("Modifier: %d\n\n", item_pointer->modifier);
                    }
                }
            }
            // If no item found in the inventory, it will print that data could not
 be found
            else
            {
                printf("\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)
        {
            struct inventory_item *item_pointer;
            item_pointer = inventory_first(inventory_pointer);
            // If function inventory_first returned the address of the first item a
llocated, then this if statement will run, it will set the index number of the item
            // into location variable, then it will call the inventory_delete funct
ion which will set this item pointer to NULL and if successfull it will return a 0
            // Then if true, the allocated memory for that item will be freed and t
his will run again in a for loop doing the same process for the next items in the i
nventory
            if (item_pointer != NULL)
            {
                delete_result = inventory_delete(inventory_pointer, item_pointer->i
tem_key);
                free(item_pointer);
                c1 = 0;
                while(c1 < MAXITEMS)
                {
                    delete_result = -2;
                    item_pointer = inventory_next(inventory_pointer);
                    if (item_pointer != NULL)
                    {
                        delete_result = inventory_delete(inventory_pointer, item_po
inter->item_key);
                        if (delete_result == 0)
                        {
                            free(item_pointer);
                        }
                    }
                    c1++;
                }
            }

            destroy_result = inventory_destroy(inventory_pointer);
            if (destroy_result == 0)
            {
                // Exist program after all dynamic memory has been freed
                exit(0);
            }
        }

        // Error message printed if unexpected first word enterd or incorrect numbe
r of arguments is entered
        else
        {
            printf("Option \"%s\" is not recognized...\n", command);
        }
    }
    return 0;
}
```