جامعـة نيويورك أبـوظـبي
NYU ABU DHABI

ENGR-UH 1000
Computer Programming for Engineers

Assignment 1-Civil Engineering Case Study
Pressure Drop of a Fluid Through a Pipe

Mohamed Eid
Fall 2024

Step 1. Problem identification:

In computer engineering, pointers and data manipulation are fundamental concepts, particularly in real-world applications such as gaming. Many video games rely on dynamic data structures to track player attributes like health, money, and progress. However, players often encounter challenges when the game's internal data becomes misaligned or incorrectly updated. For example, insufficient health or money can create frustrating obstacles, leading to poor user experiences. These issues are typically tied to how game data is accessed, modified, or managed in memory, particularly when pointers to critical game variables are improperly handled or corrupted.

Step 2. Gathering information and input/output description:

Relevant information:

To address the challenges in game data manipulation, tools like Cheat Engine are commonly used. These tools access and modify the game's memory by using pointers to locate and alter specific data, such as health or money values.

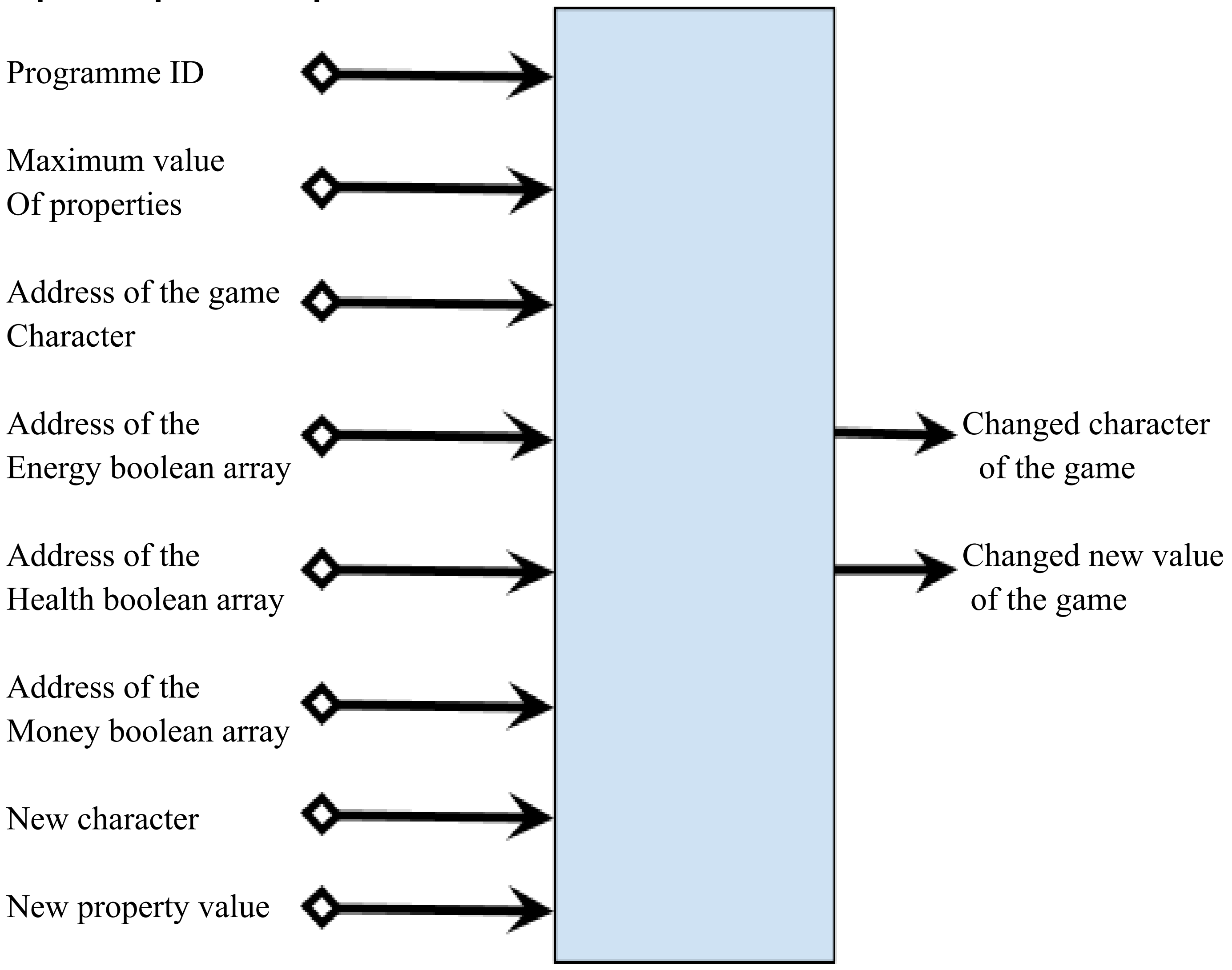
The software provides the user with three options. Option(1) lets the user change the UI character of the game. Option(2) gives the user the option to change the value of each property of the game. Option(3) exits the user and ends the game hacking.

When the user selects option(1), the software shows the current character of the game and prompts for a new character. This hacks the game and changes its character.

When the user selects option(2), the software prompts the user to choose from the three properties: health, money, and energy. This lets the user hack the only one property of the game. After the user chooses from the three properties, the software shows the current value of the property the user chose and prompts him/her to enter a new value for it. Next, the software prompts the user if he/she wants to modify the value he/she provided. If the user chooses to modify the value he/she provided, the user will be taken back to change the values. But if the user does not want to modify the value he/she provided, the software will change the previous value of the property with the new value the user provided.

When the user selects option(3), the software will terminate.

Input/Output description:



Programme ID
Maximum value Of properties
Address of the game Character
Address of the Energy boolean array
Address of the Health boolean array
Address of the Money boolean array
New character
New property value
Changed character of the game
Changed new value of the game

The software reads values from the gameinfo file such as the programme ID, maximum value of properties, address of the game character, the address of energy boolean array, address of the health boolean array, address of the mone boolean array, accepts the new character and new value for the game property from the user. Using these input values, the software lets the user the character and values of the game.

Step 3. Design of algorithm and hand-solved problems:

Design of algorithm:

```
// Step 1: Open the input file "gameinfo.txt"
OPEN file "gameinfo.txt" for reading as fileread
IF file open fails
   PRINT "Error"
   RETURN -1

// Step 2: Declare variables
DECLARE processId as Integer
DECLARE amoneyPtr, ahealthPtr, aenergyPtr, auiCharPtr as Long Integer
DECLARE maxProp as Integer
DECLARE newUiChar, choice as Character

// Step 3: Read values from the file
READ processId from fileread
READ maxProp from fileread
READ amoneyPtr from fileread in hexadecimal format
READ ahealthPtr from fileread in hexadecimal format
READ aenergyPtr from fileread in hexadecimal format
READ auiCharPtr from fileread in hexadecimal format

// Step 4: Attach the program to the target game process
CALL attachToProcess(processId)

// Step 5: Declare and initialize a properties array
DECLARE propertiesArray as Array of Long Integer with size 3
propertiesArray[0] = amoneyPtr
propertiesArray[1] = ahealthPtr
propertiesArray[2] = aenergyPtr

// Step 6: Allocate a 2D array for game properties
DECLARE gameprops as 2D Array of Long Integer with 3 rows
FOR each row i from 0 to 2
   CREATE new array of size maxProp for gameprops[i]
   INITIALIZE gameprops[i] with 0 values

// Step 7: Display the menu and get user choice
PRINT menu options for "UI Hacking Mode", "Data Hacking Mode", "Exit"
READ user choice into variable choice

// Step 8: Process user choice
```

```
    SWITCH choice
        CASE '1'  // UI Hacking Mode
            PRINT current character by calling readUIChar(processId, auiCharPtr)
            PRINT "Enter new UI character"
            READ newUiChar
            CALL writeUIChar(processId, auiCharPtr, newUiChar)
            PRINT success message

        CASE '2'  // Data Hacking Mode
            CALL dataHackFunc(gameprops, propertiesArray, processId, maxProp)

        CASE '3'  // Exit
            PRINT "You have chosen to exit the program."
            PRINT "Thank you!"

        DEFAULT  // Invalid choice
            PRINT "Invalid choice. Please select 1, 2, or 3."

    // Step 9: Close the file
    CLOSE fileread

    // Step 10: Free allocated memory for game properties
    FOR each row i from 0 to 2
        DELETE gameprops[i]
    DELETE gameprops

    // Step 11: Detach from the game process
    CALL detachProcess(processId)


// Function: dataHackFunc
FUNCTION dataHackFunc(gameprops, propertiesArray, processId, maxProp)
    DECLARE propNumber as Integer
    DECLARE newValue as Integer
    DECLARE count, ctr as Integer
    DECLARE array as array of strings

    WHILE true
        // Step 1: Select game value to hack
        propNumber = -1
    newValue = -1
    count = 0
    ctr = 0

    // Step 2: Ask user to select a property (Money, Health, or Energy)
    WHILE propNumber is invalid (less than 0 or greater than 2)
        IF count == 0 THEN
            PRINT "Select which game value to hack"
        ELSE
            PRINT "Invalid choice. Please choose 0, 1, or 2."
        PRINT options for Money, Health, Energy
        READ propNumber

        INCREMENT count

    // Step 3: Count the values for the selected property
    FOR i from 0 to maxProp
        value = readValue(processId, propertiesArray[propNumber] + i)
        IF value >= 1 THEN
            INCREMENT ctr

    // Step 4: Get new value to set
    WHILE newValue is invalid (less than 0 or greater than maxProp)
        IF count == 0 THEN
            PRINT "Current value: ctr"
            PRINT "Enter new value (between 0 and maxProp)"
        ELSE
            PRINT "Invalid input. Enter a value between 0 and maxProp."
        READ newValue

        INCREMENT count
        DECLARE c as Character
    // Step 5: Ask user if they want to modify values before hacking
    PRINT "Do you want to modify values before hacking? (n for no, any key for yes)"
    READ c

    IF c == 'n'
        BREAK

// Step 6: Modify values based on the newValue
```

```
    IF newValue == 0 THEN
        FOR i from 0 to maxProp
            CALL modifyValue(processId, propertiesArray[propNumber] + i, 0)
    ELSE
        FOR i from 0 to newValue
            CALL modifyValue(processId, propertiesArray[propNumber] + i, 1)
PRINT "You have successfully changed the", array[propNumber], " from ", ctr, " to ", newValue
```

Hand solved problems:

Test case-1(option 1, changing the character of the game)

```
Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit
1
This is the current character of the game: *
What do you want to change it to?
+
You have successfully changed the character from * to +
You have successfully hacked the game
```

Test case-2(option 2, changing the value of the money property of the game)

```
Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit
2
Select which game value to hack
[0] Money
[1] Health
[2] Energy
0
This is the value at the moment: current_value_of_money
```

```
Enter the number you want to change your game value to(0 to maximum_value_of_the_properties)
0
Do you want to modify it before hacking? n for no and any other key for yes
n
You have successfully the money property of the game from current_value_of_money to new_value
```

Test case-3(option 2, changing the value of the health property of the game)

```
Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit
2
Select which game value to hack
[0] Money
[1] Health
[2] Energy
1
This is the value at the moment: current_value_of_health
Enter the number you want to change your game value to(0 to maximum_value_of_the_properties)
0
Do you want to modify it before hacking? n for no and any other key for yes
n
You have successfully the health property of the game from current_value_of_health to new_value
```

Test case-4(option 2, changing the value of the energy property of the game)

```
Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit
2
Select which game value to hack
[0] Money
[1] Health
```

[2] Energy

2

This is the value at the moment: current_value_of_energy

Enter the number you want to change your game value to(0 to maximum_value_of_the_properties)

0

Do you want to modify it before hacking? n for no and any other key for yes

n

You have successfully the health property of the game from current_value_of_energy to new_value

Test case-5(option 3, exiting the software)

You have chosen to exit the software

Thank you!

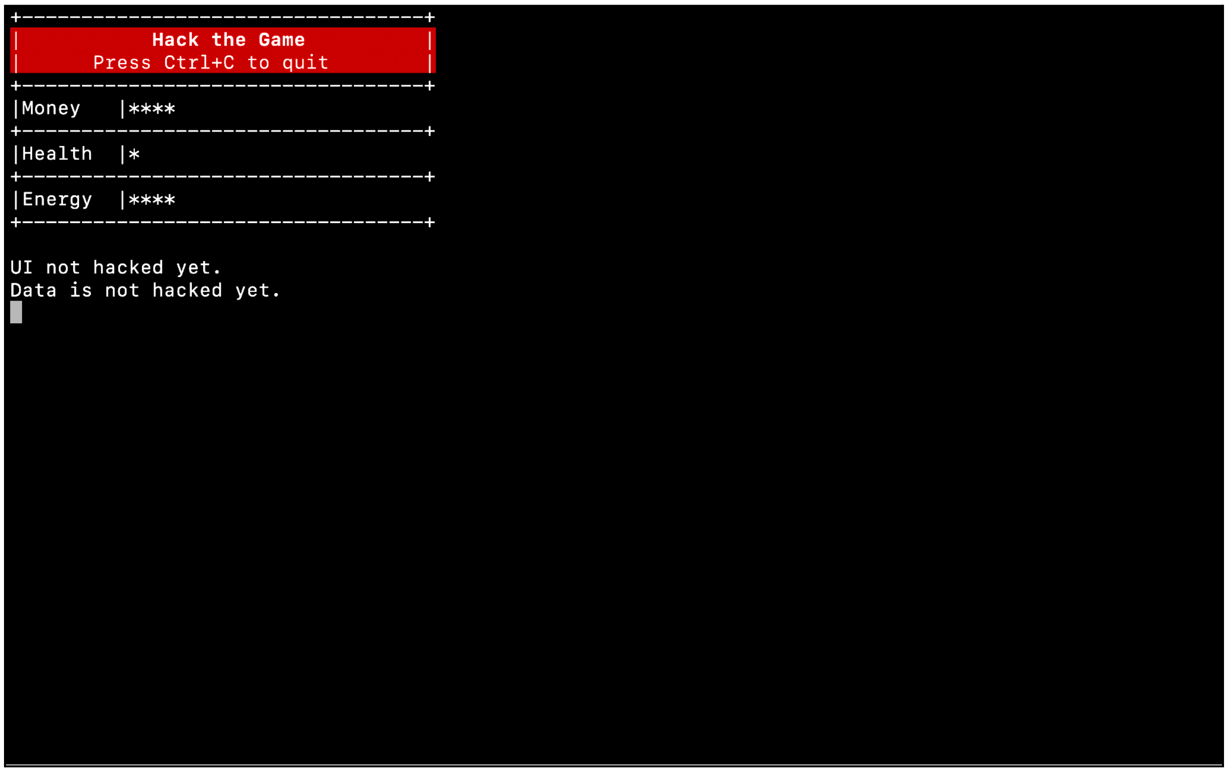
Step 4. Implementation

```
#include <iostream>
#include <fstream>
#include "hacking_functions.h"
void dataHackFunc(long**, long[], int, int);

using namespace std;
int main(){
   //create an input file stream and open the gameinfo.txt file
   ifstream fileread("gameinfo.txt");
   if(fileread.fail()){
       cout<<"Error";
       return -1;
   }
   int processId=0;
   // long  *healthPtr, *energyPtr, *uiCharPtr;
   // long *moneyPtr;
   long amoneyPtr, ahealthPtr, aenergyPtr, auiCharPtr;
   int maxProp=0;
   char newUiChar, choice;
   //read the content of the file one line at a time
   fileread>>processId;
   fileread>>maxProp;
   //read the address value in hexadecimal form
   fileread>>hex>>amoneyPtr;
   fileread>>hex>>ahealthPtr;
   fileread>>hex>>aenergyPtr;
   fileread>>hex>>auiCharPtr;

   //attach the program to the game
   attachToProcess(processId);

   long propertiesArray[3]={amoneyPtr, ahealthPtr, aenergyPtr};

   //create a 2d array and populate it with values that read from the gameinfo.txt file
   long **gameprops=new long*[3];

   for(int i=0; i<3; i++){
       gameprops[i]=new long[maxProp];
       for(int j=0; j<maxProp; j++){
           gameprops[i][j]=0;
        }
   }

       cout<<"\n\nPlease select the desired mode"<<endl;
       cout<<"[1] UI Hacking Mode"<<endl;
       cout<<"[2] Data Hacking Mode"<<endl;
       cout<<"[3] Exit"<<endl<<endl;
       cin>>choice;
       switch (choice)
       {
       case '1':
           //code to change the ui character of the game
           cout<<"This is the current character of the game: "<<readUIChar(processId,
auiCharPtr)<<endl;
           cout<<"What do you want to change it to?"<<endl;
           cin>>newUiChar;
           //calling the predefined function "writeUIchar" to change the character of the game
           cout<<"You have successfuly changed the character from "<<readUIChar(processId,
auiCharPtr)<<" to "<<newUiChar<<endl;
           writeUIChar(processId,  auiCharPtr, newUiChar);
           break;
       case '2':
           //code to modify the value inside the game
           dataHackFunc(gameprops, propertiesArray, processId, maxProp);
           break;
       case '3':
           cout<<"You have chosen to exit the program."<<endl;
           cout<<"Thank you!"<<endl;
           break;
       default:
           //code whenever the user enters other than 1, 2, or 3
           cout<<"You entered a wrong key"<<endl;
           break;
       }
   //close the file
   fileread.close();
   //iterate over and delete the memory address three arrays are holding
   for(int i=0; i<3; i++){
       delete[]gameprops[i];
   }
   //delete the memory address of the array that contains the three arrays
   delete []gameprops;
   detachProcess(processId);
}

void dataHackFunc(long **gamePropsArg, long propertiesArray[], int pID, int max){
   int propNumber=-1, newValue=-1, count=0, ctr=0;
   string array[3]={"Money", "Health", "Energy"};
   while(true){
       propNumber=-1; newValue=-1; count=0; ctr=0;
       while(propNumber<0 || propNumber>2){//as long as the user enters a wrong key
           if(count==0){//if the first time the user is attempting
               cout<<"Select which game value to hack"<<endl;
           }
           else{//or if the user tried it before and entered a wrong key
               cout<<"Wrong choice, choose 0, 1, or 2"<<endl;
           }
           cout<<"[0] Money"<<endl;
           cout<<"[1] Health"<<endl;
           cout<<"[2] Energy"<<endl;
           cin>>propNumber;
           count++;//keep track of how many times the user entered this block
       }
       count=0;//make count 0 so that the other blocks can use it for the same logic

       for(int i=0; i<max; i++){
           int a=readValue(pID, propertiesArray[propNumber]+i);
           if(a>=1){
               ctr++;
           }
       }



       while(newValue<0 || newValue>max){
           if(count==0){
               cout<<"This is the value at the moment: "<<ctr<<endl;
               cout<<"Enter the number you want to change your game value to (0 to 
"<<max<<")"<<endl;
           }
           else{
               cout<<"Wrong choice, choose from 0 through "<<max<<endl;
           }
           cin>>newValue;
           count++;
       }
       char c;
       cout<<"Do you want to modify values before hacking? n for no and any other key for 
yes"<<endl;
       cin>>c;
       if(c=='n'){
       break;
       }
   }



   if(newValue==0){
       for(int i=0; i<max; i++){
           modifyValue(pID, propertiesArray[propNumber]+i, 0);
       }
   }

   else{
       for(int i=0; i<newValue; i++){
           //using the predefined function "modifyValue" modify the game memory
           modifyValue(pID, propertiesArray[propNumber]+i, 1);
       }
   }
   cout<<"You have successfuly changed the " <<array[propNumber]<<" property from "<<ctr<<" to 
"<<newValue<<endl;
```

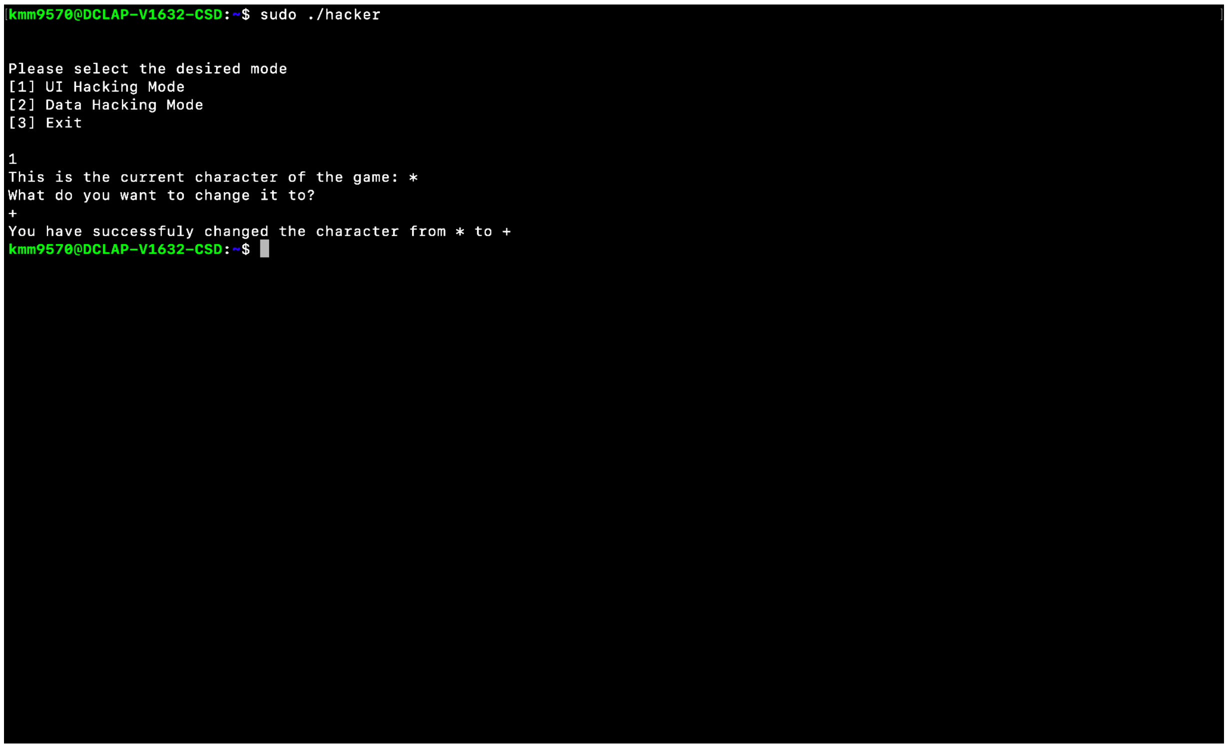
```
}
```

Step 5. Software testing and verification

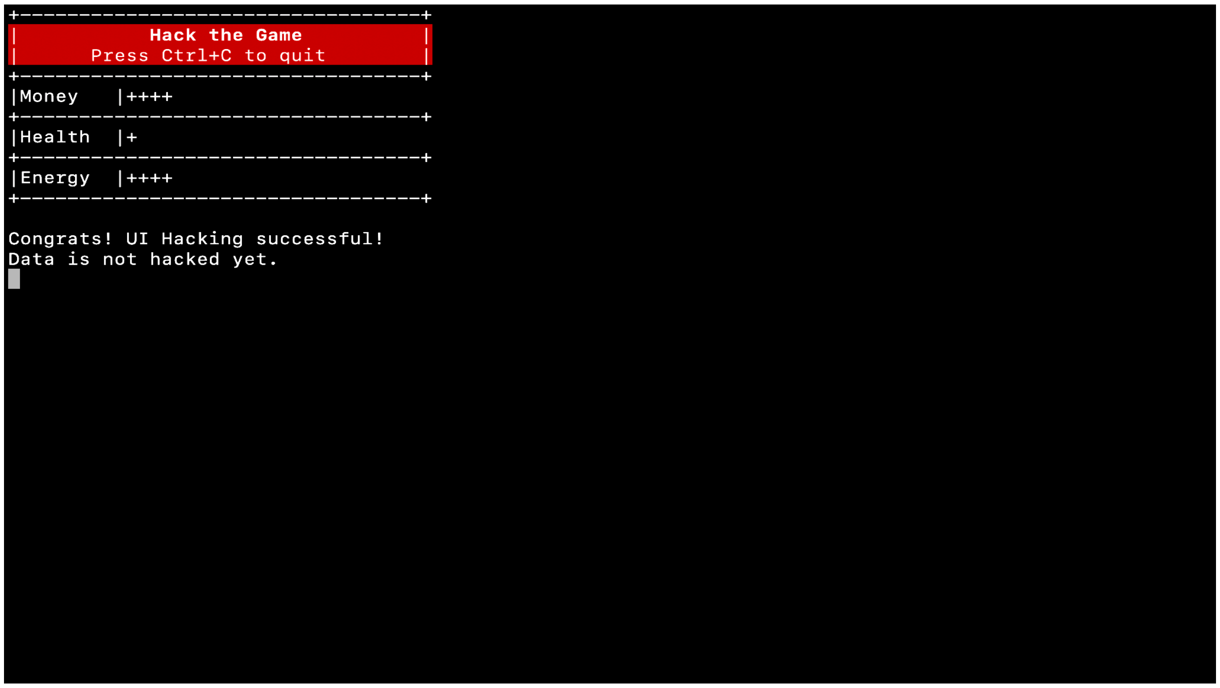
Test case-1(option 1, changing the character of the game):



```
+-----------------------------------+
|           Hack the Game           |
|       Press Ctrl+C to quit        |
+-----------------------------------+
|Money    |****
+-----------------------------------+
|Health   |*
+-----------------------------------+
|Energy   |****
+-----------------------------------+

UI not hacked yet.
Data is not hacked yet.
```

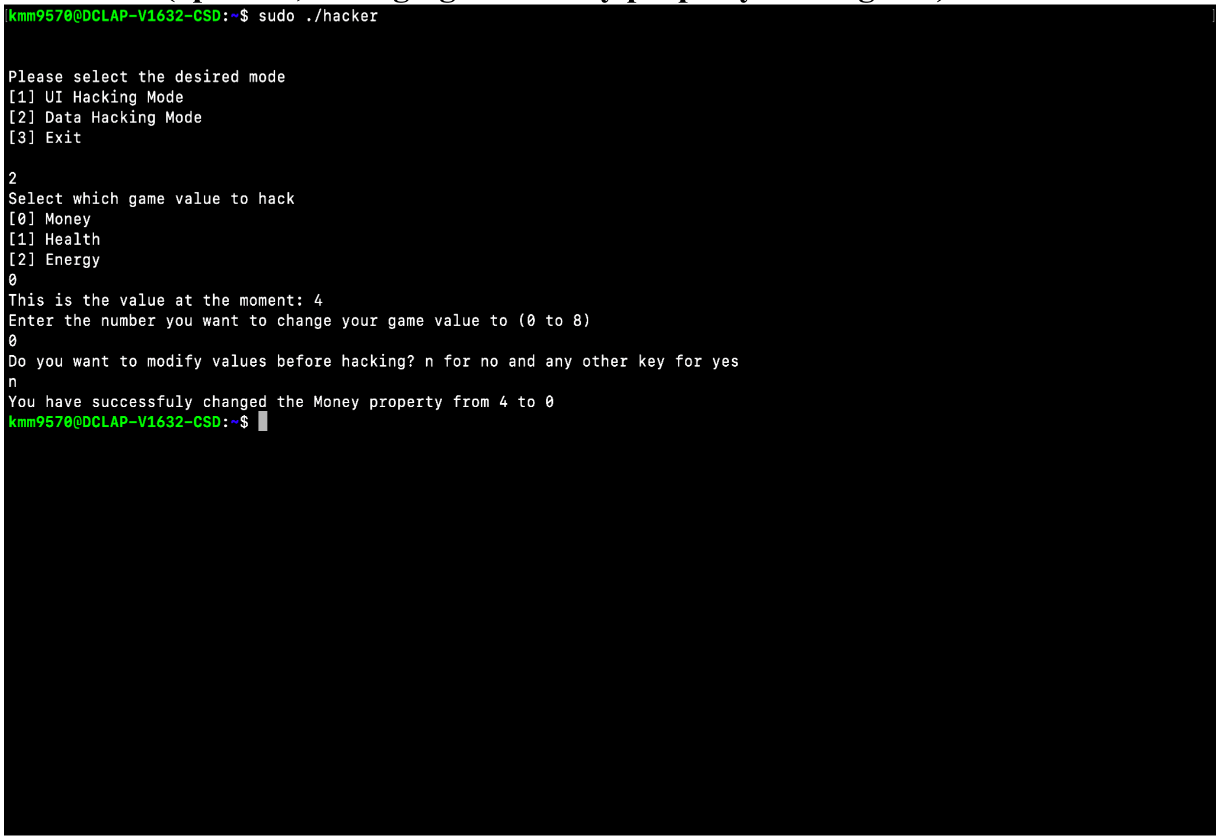


```
kmm9570@DCLAP-V1632-CSD:~$ sudo ./hacker


Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit

1
This is the current character of the game: *
What do you want to change it to?
+
You have successfuly changed the character from * to +
kmm9570@DCLAP-V1632-CSD:~$
```

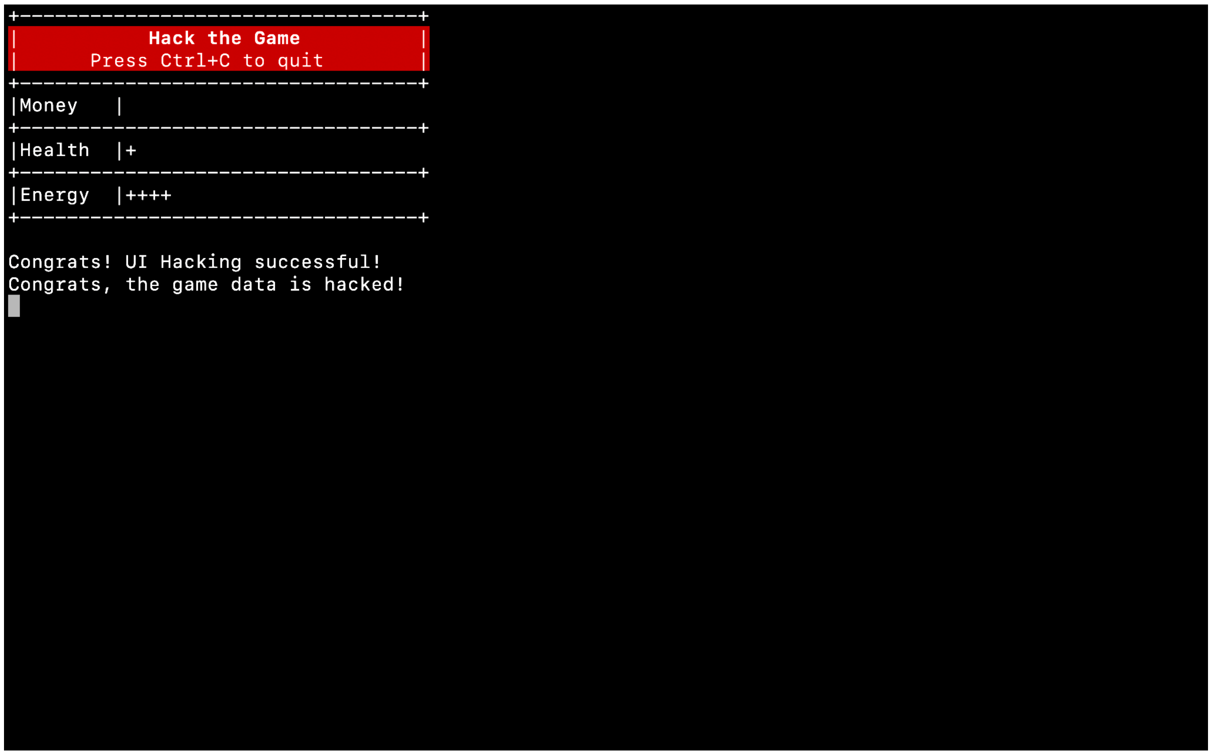


```
+-----------------------------------+
|           Hack the Game           |
|       Press Ctrl+C to quit        |
+-----------------------------------+
|Money    |++++
+-----------------------------------+
|Health   |+
+-----------------------------------+
|Energy   |++++
+-----------------------------------+

Congrats! UI Hacking successful!
Data is not hacked yet.
```

Test case-2(option 2, changing the money property of the game)

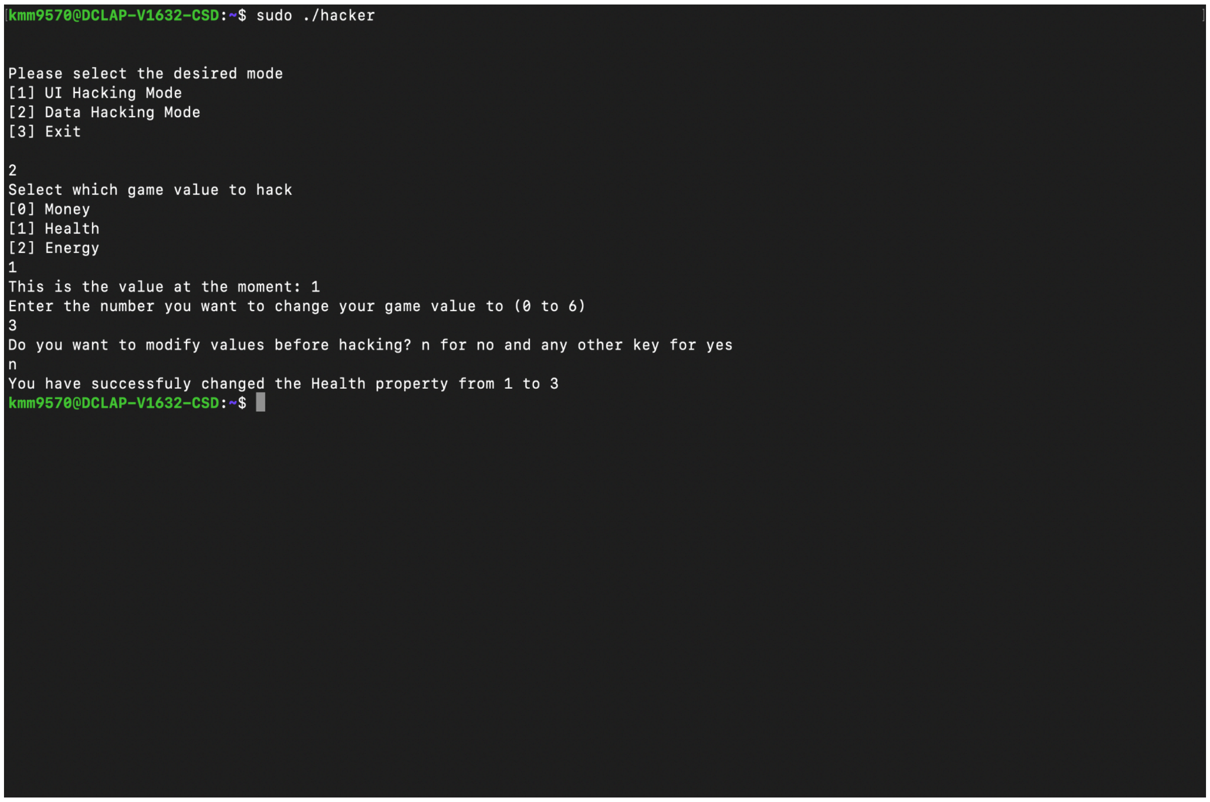


```
kmm9570@DCLAP-V1632-CSD:~$ sudo ./hacker


Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit

2
Select which game value to hack
[0] Money
[1] Health
[2] Energy
0
This is the value at the moment: 4
Enter the number you want to change your game value to (0 to 8)
0
Do you want to modify values before hacking? n for no and any other key for yes
n
You have successfuly changed the Money property from 4 to 0
kmm9570@DCLAP-V1632-CSD:~$
```

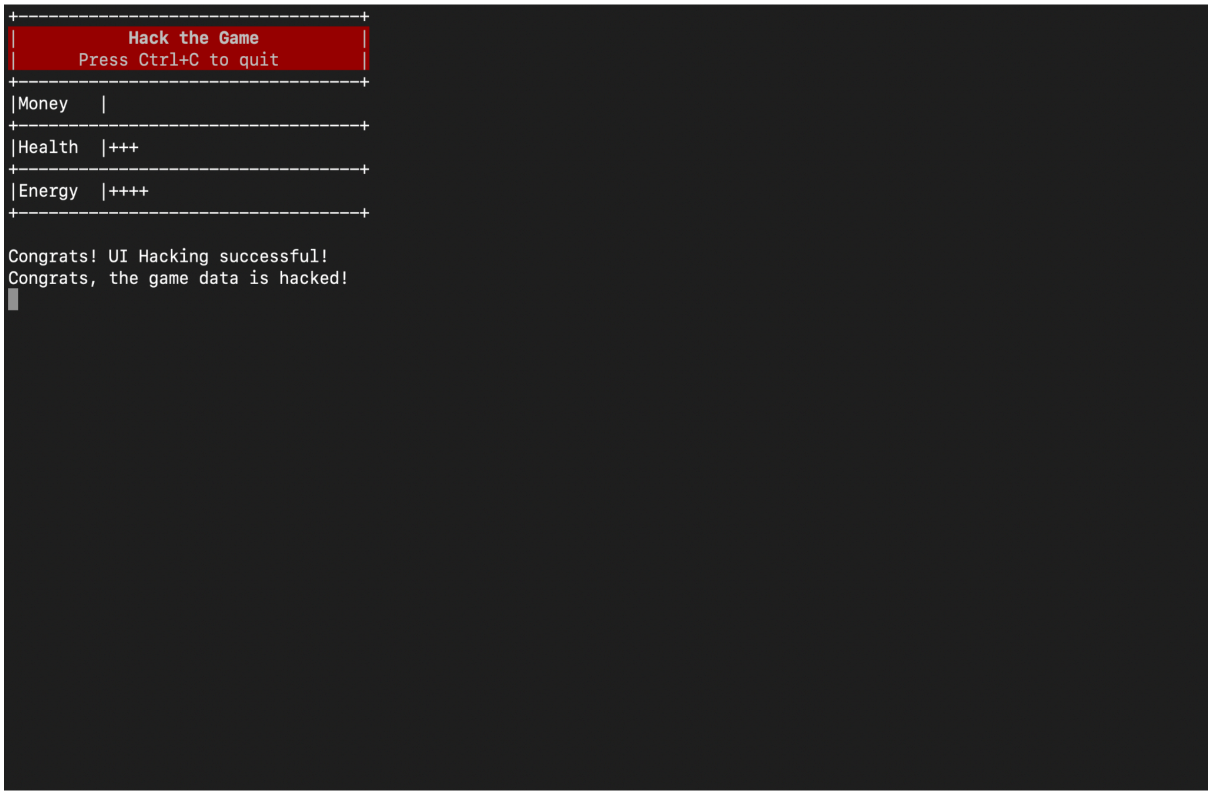


```
+------------------------------------+
|            Hack the Game           |
|       Press Ctrl+C to quit         |
+------------------------------------+
|Money    |
+------------------------------------+
|Health   |+
+------------------------------------+
|Energy   |++++
+------------------------------------+

Congrats! UI Hacking successful!
Congrats, the game data is hacked!
```

Test case-3(option 2, changing the health property of the game)

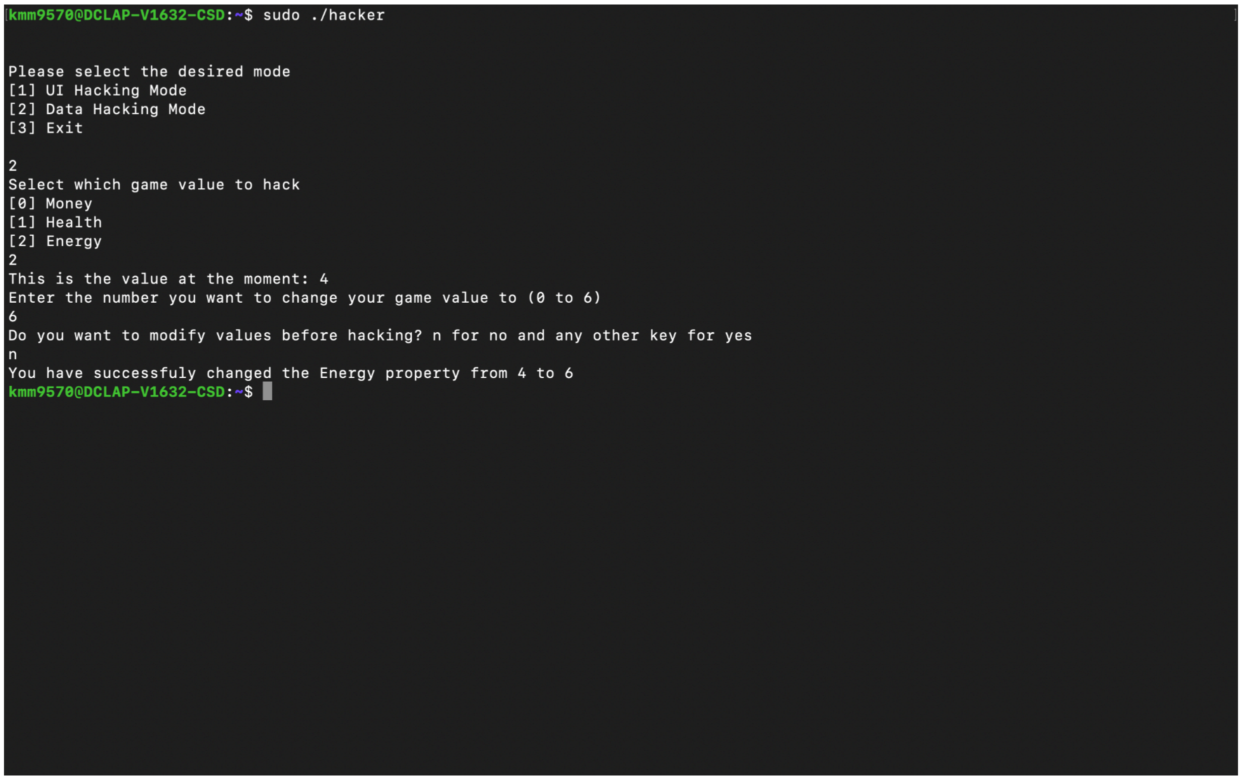


```
kmm9570@DCLAP-V1632-CSD:~$ sudo ./hacker


Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit

2
Select which game value to hack
[0] Money
[1] Health
[2] Energy
1
This is the value at the moment: 1
Enter the number you want to change your game value to (0 to 6)
3
Do you want to modify values before hacking? n for no and any other key for yes
n
You have successfuly changed the Health property from 1 to 3
kmm9570@DCLAP-V1632-CSD:~$
```

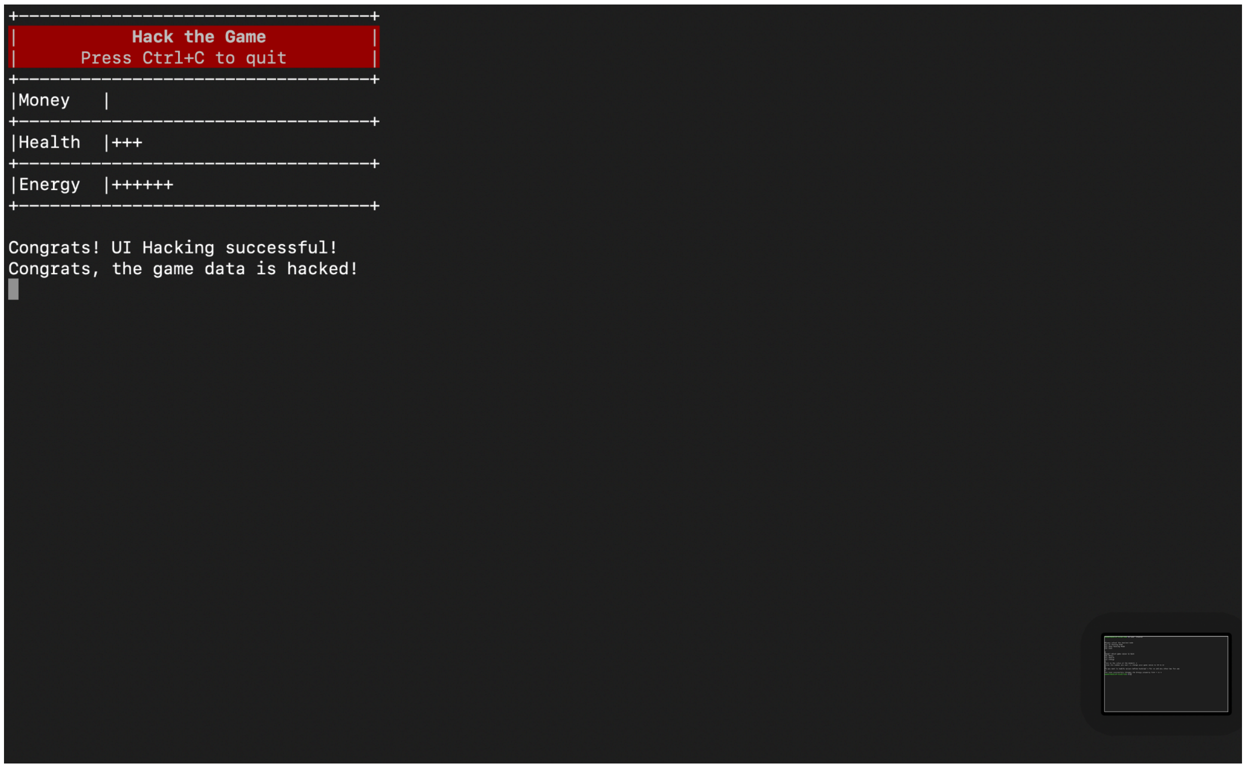


```
+------------------------------------+
|           Hack the Game            |
|      Press Ctrl+C to quit          |
+------------------------------------+
|Money    |
+------------------------------------+
|Health   |+++
+------------------------------------+
|Energy   |++++
+------------------------------------+

Congrats! UI Hacking successful!
Congrats, the game data is hacked!
```

Test case-4(option 2, changing the energy property of the game)

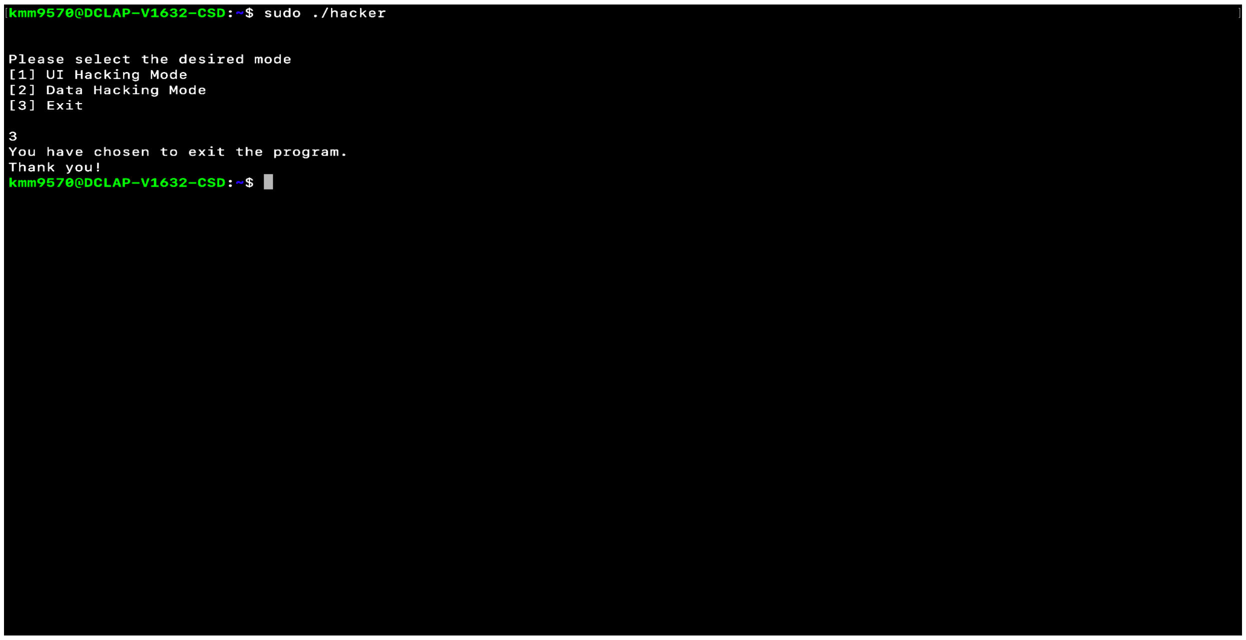


```
kmm9570@DCLAP-V1632-CSD:~$ sudo ./hacker


Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit

2
Select which game value to hack
[0] Money
[1] Health
[2] Energy
2
This is the value at the moment: 4
Enter the number you want to change your game value to (0 to 6)
6
Do you want to modify values before hacking? n for no and any other key for yes
n
You have successfuly changed the Energy property from 4 to 6
kmm9570@DCLAP-V1632-CSD:~$
```



```
+-----------------------------------+
|           Hack the Game           |
|       Press Ctrl+C to quit        |
+-----------------------------------+
|Money    |
+-----------------------------------+
|Health   |+++
+-----------------------------------+
|Energy   |++++++
+-----------------------------------+

Congrats! UI Hacking successful!
Congrats, the game data is hacked!
```

Test case-5(option 3, exiting the software



```
kmm9570@DCLAP-V1632-CSD:~$ sudo ./hacker


Please select the desired mode
[1] UI Hacking Mode
[2] Data Hacking Mode
[3] Exit

3
You have chosen to exit the program.
Thank you!
kmm9570@DCLAP-V1632-CSD:~$
```

User Guide

To use this program, the user should first ensure they are running it in a **Linux-based environment** (preferably Ubuntu) to avoid conflicts with security software on other operating systems. Upon running the program, the user is presented with a menu offering three options: **UI Hacking Mode**, **Data Hacking Mode**, and **Exit**. In **UI Hacking Mode**, the user can view and modify the in-game UI character by selecting a new character. In **Data Hacking Mode**, the user selects a game property (money, health, or energy) and modifies its value by entering a new number. The program will read the current values from the game's memory and allow the user to make changes. The program then applies these changes to the game in real-time, providing the user with a customized gaming experience. After using the program, the user can safely exit, at which point all dynamically allocated memory is cleaned up and the program detaches from the game process. The user can find this proposal and the C++ source code in the following github repository:
https://github.com/Kumneger49/Hack-my-game