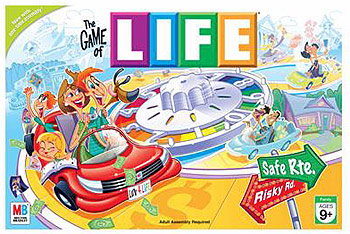
PROGRAM 3 - BRUMMETTLAND GAME

(Mixture between Candyland and Life Created by April Crockett Modified By Travis Brummett)



# Assignment & Due Dates

* **Assignment Date:** Monday, October 7, 2019
* **Due Date:** Wednesday, October 23, 2019

# Description

You are writing a game where your players navigate through an imaginary world (board) using an imaginary dice.

Your imaginary world has good things and bad things in it.

Each player begins the game with zero score and are on board space zero. Each player has a turn where all they do is roll a dice.

When a player lands on a space, either good or bad is selected randomly. If good, then something good happens to the player, a smiley face is printed, and points are added to their score. If bad, then something bad happens to the player, a frown face is printed, and points are subtracted from their score.

The game ends when a player lands on or goes past the ending space on the board.

# Constant Variables

* Maximum Number of Players possible: 20
* Number of Spaces on Board: 25

# Overall Program

* You will have three arrays:
  + names – will hold the names of players playing the game. Example: names[0] could hold Travis Brummett and names[1] could hold Jane Jetson if Travis and Jane are currently playing the game.
  + boardSpace – will hold the current location (space) of players playing the game. Example: boardSpace[0] could hold 15 and boardSpace[1] could hold 6 if Travis was currently at space 15 on the board and Jane was currently at space 6 on the board.
  + score –will hold how much score each player has playing the game. Example: score[0] could hold 50000 and score[1] could hold -4000 if Travis currently had 50,000 points and Jane had -4,000. (yes – people can have a negative score in the game!)
* You will allow your players to play the game as many times as they want
* You can name your imaginary land whatever you want.
* You will create two text files – bad.txt and good.txt. There is more details on this later in this document.

# PROGRAM SPECIFICATIONS / flow

## Main Function

* Print out the name of your imaginary land/game. (Mine is called BRUMMETTLAND – but you can name yours whatever you want)
* Ask user how many players are playing and validate that they didn’t enter more than the maximum number of players – and that it is more than zero.
* Set up your arrays for the start of the game! Set each element of names array to an empty string, each element of boardSpace array to zero, and each element of score array to 0.0.
* Call the **getPlayersNames**() function
* Create a loop that will iterate until one of the players has reached the last space on the board.
* Create another loop to iterate through each of the players for each turn. For each player’s turn, you will:  
  + Call the **RollDice**() function
  + Add the dice number to the position in the boardSpace array that corresponds to the current player.
  + If they have reached the end of the board, then call the **PlayerFinishedBoard**() function.
  + If they have not reached the board, then call the **ActivateActionOnSpace**() function.
* After a player reached the ending space on the board and the results have all been printed, then ask the user if they want to play again.

## getPlayerNames Function

**Parameters:** string array (containing player’s names), and an integer holding the number of players that want to play

**Return data:** none

Get each player’s name and place their name in each element of the names array. Notice when the player’s names are printed out in the samle output. Your program should print out the player’s names in the same way.

## rollDice() Function

**Parameters:** a string, holding the current player’s name

**Return data:** integer, with the dice number rolled

Print out the player’s name and tell them to press enter to roll die.

Randomly generate a number between 1 & 6 (for the dice roll)

Print a cool picture of the dice that was rolled (code given to you in **dice.cpp)**

## PlayerFinishedBoard Function

**Parameters**: all the arrays, the number of players

**Return data**: none

This code should execute when one of the players reached the last space on the board. It will display the results of the game – who reached the end of the board and also who ended up with the most score.

* Use a for loop to iterate through the boardSpace array and find the player who reached the highest boardSpace. Capture this player’s name.
* Print out the player who had the highest board space.
* Use a for loop to iterate through the score array and find the player who had the most points at the end of the game. Capture this player’s name.
* Print out the player who had the highest score.

## ActivateActionOnSpace Function

**Parameters**: all the arrays, the number of players, the integer index of the current player

**Return data**: none

Print out their current board space and then display what happens on the space – good or bad.

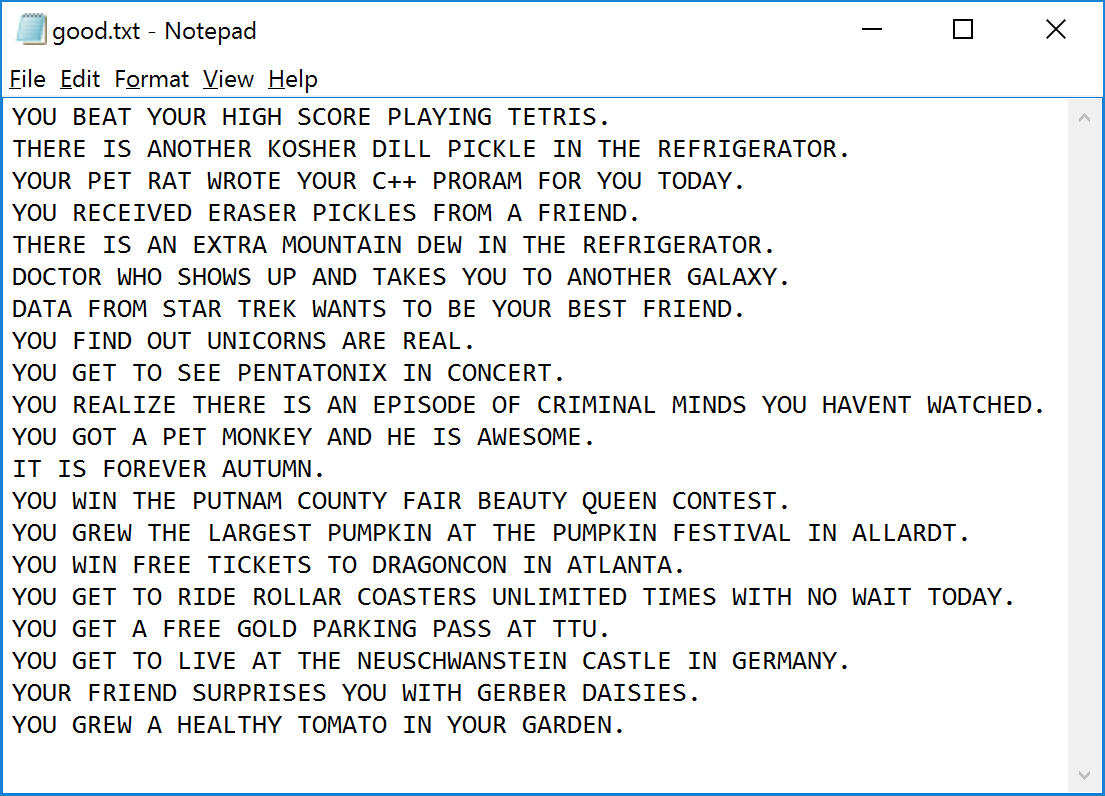
There are two text files containing the text of what happens. You will have to create each text file by opening notepad or another editor and manually typing them. The **good.txt** text file should contain 20 good things that can happen to the player. The **bad.txt** text file should contain 20 bad things that can happen to the player.

* First, randomly generate either a 1 or 2. 1 will represent something good happening and 2 will represent something bad happening.
* If good, then…
  + Open the good.txt file
  + Print out a big smiley face (doesn’t have to look exactly like mine but it has to be happy)
  + Print the word “GREAT! “ (or something similar)
* If bad, then…
  + Open the bad.txt file
  + Print out a big frowny face (doesn’t have to look exactly like mine but it has to be sad)
  + Print the word “OH NO! “ (or something similar)
* Then randomly generate a number between 1 & 20.
* Use a for loop that iterates the number of times that was randomly generated in the step above.
  + In the for loop, read in a line from the file using getline.
* After the for loop, print out the line from the file
* Then, randomly generate a number between 1 & 100,000.
  + If good, then add this randomly generated amount to this player’s score (in the score array).
  + If bad, then subtract this randomly generated amount from this player’s score (in the score array).
* Print out to the screen what score this player has now.

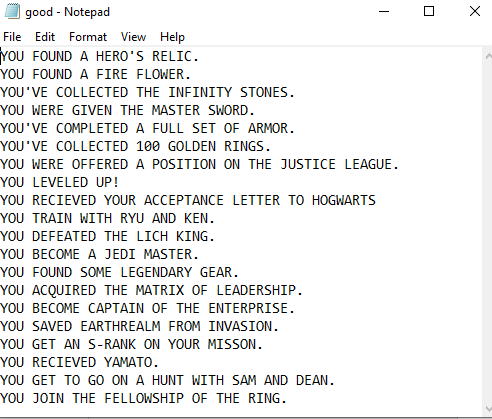
# OUTPUT – your output should look very similar to mine!!

## SAMPLE TEXT FILE – good.txt

I have provided a sample good.txt file so you can see what it looks like but you must write your own 20 “good” sentences of things that could happen in your game that would be considered good fortune for your player.

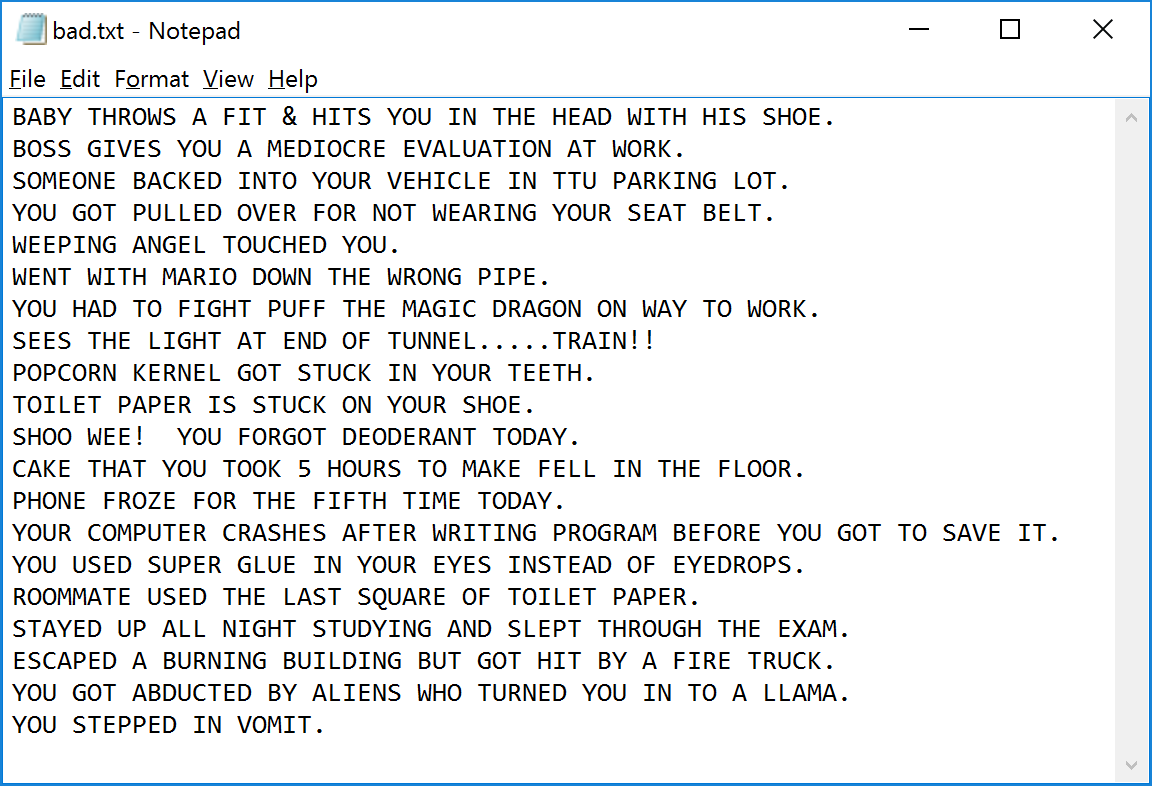


Or

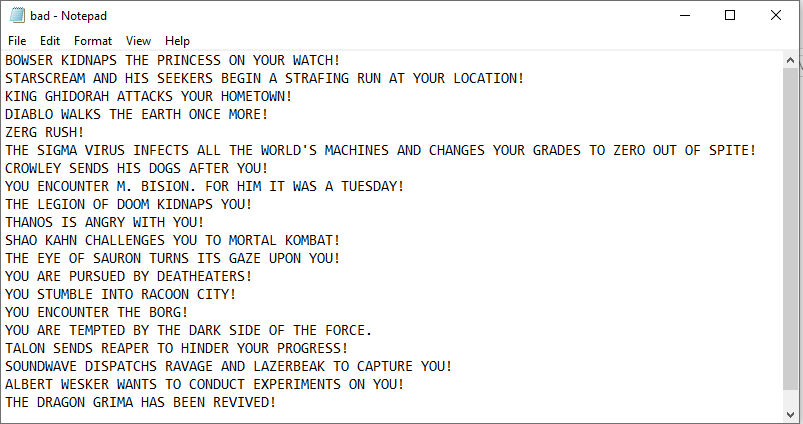


## SAMPLE TEXT FILE – bad.txt

I have provided a sample bad.txt file so you can see what it looks like but you must write your own 20 “bad” sentences of things that could happen in your game that would be considered good fortune for your player.

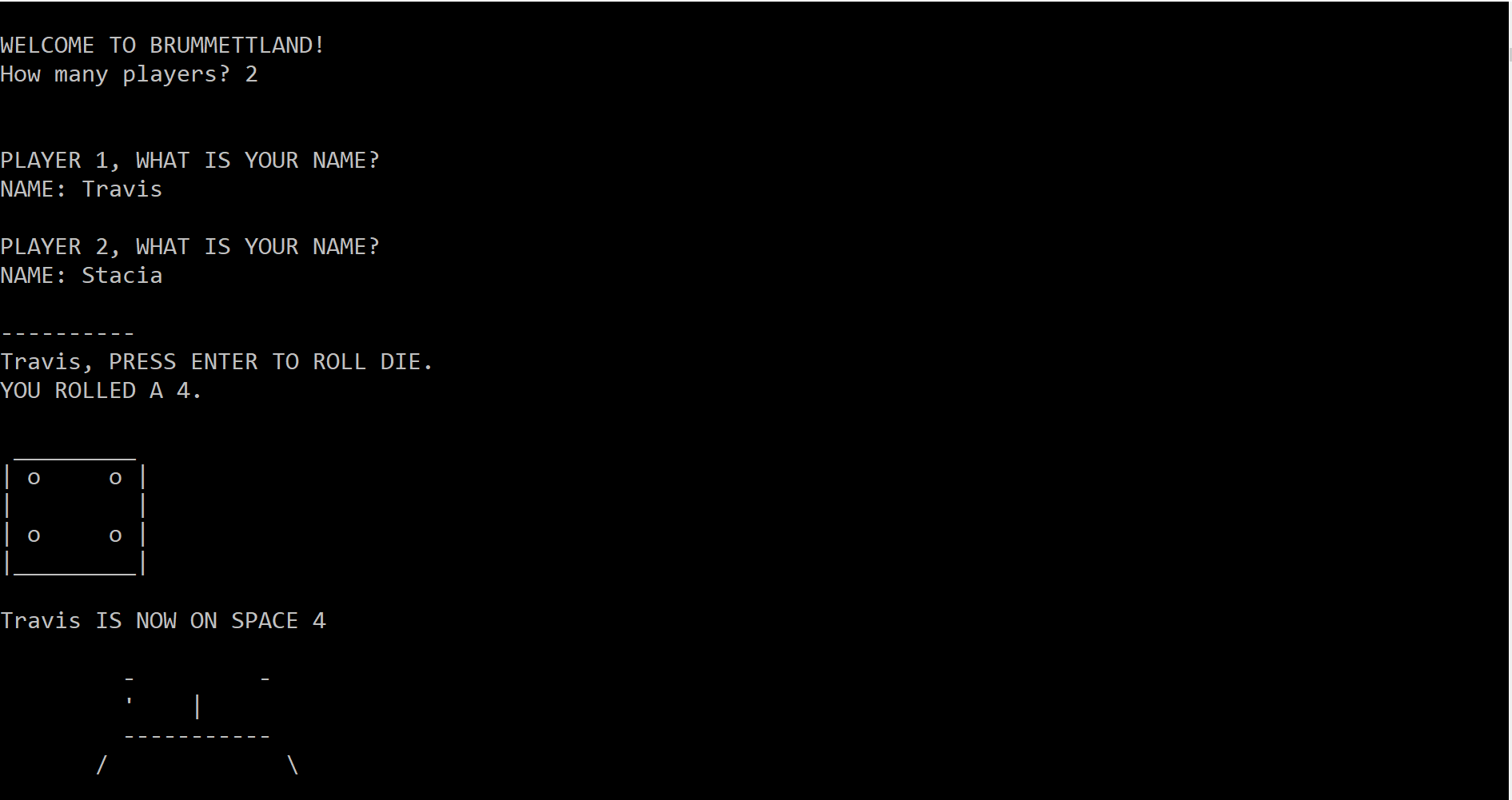


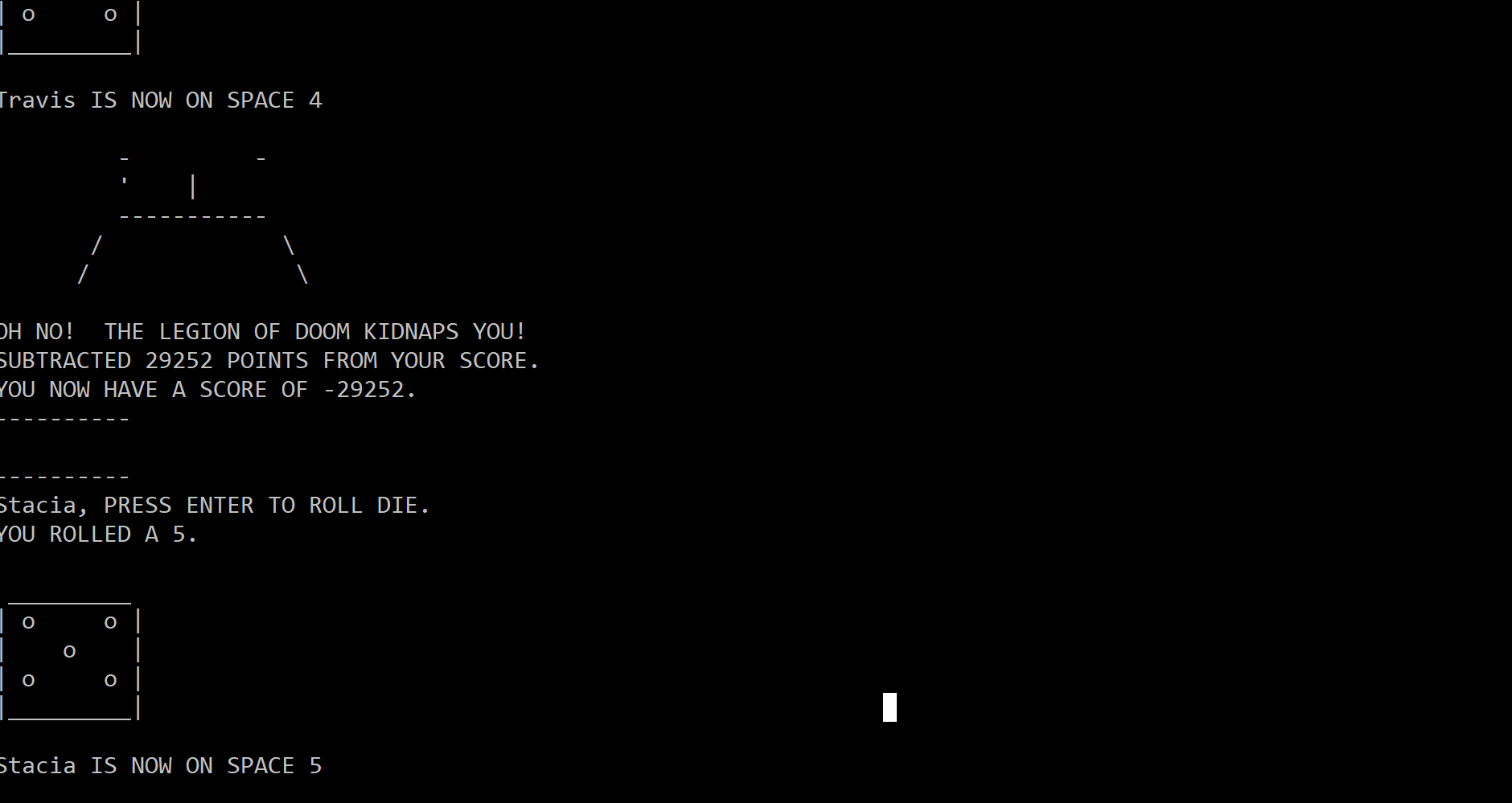
Or

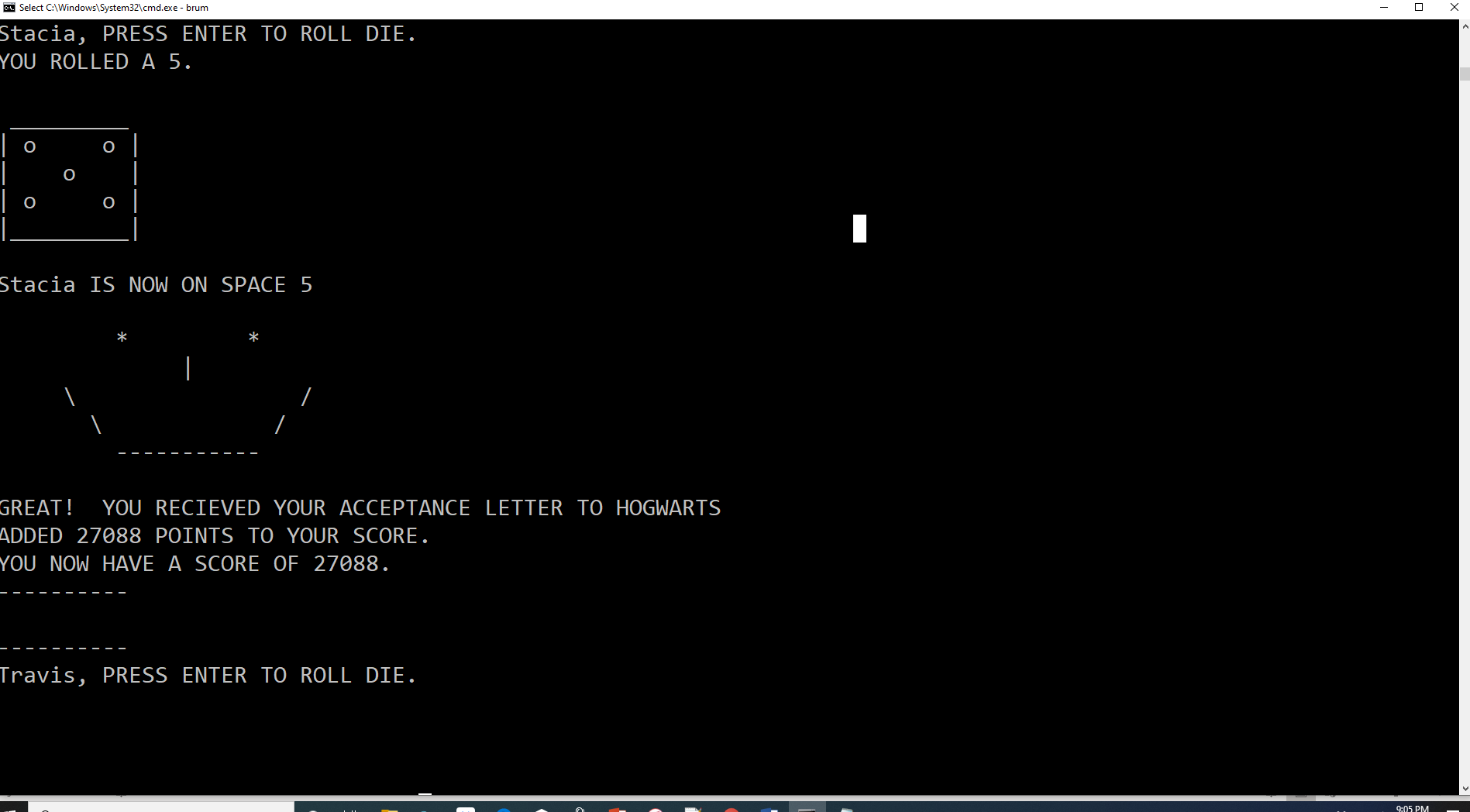
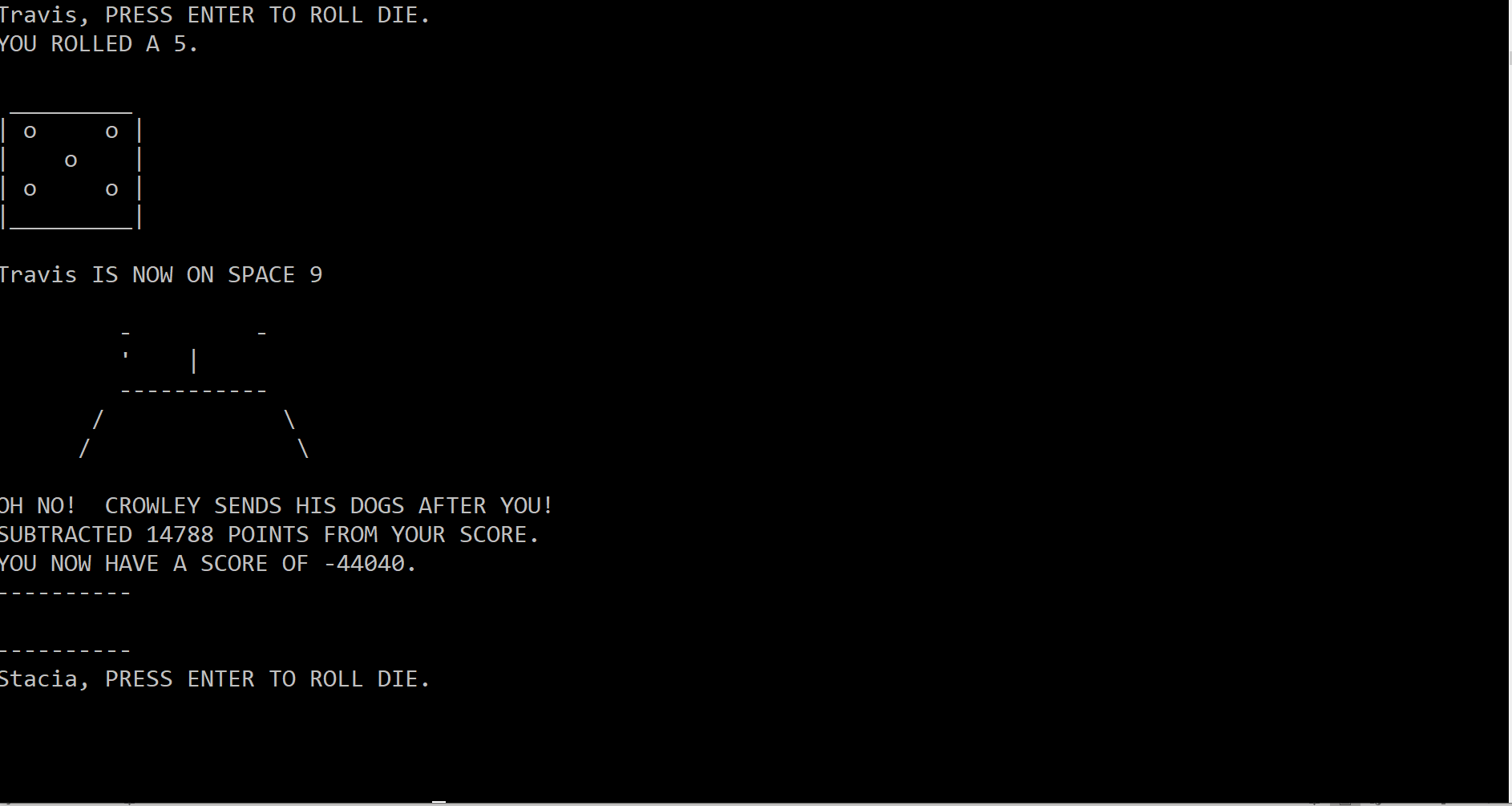


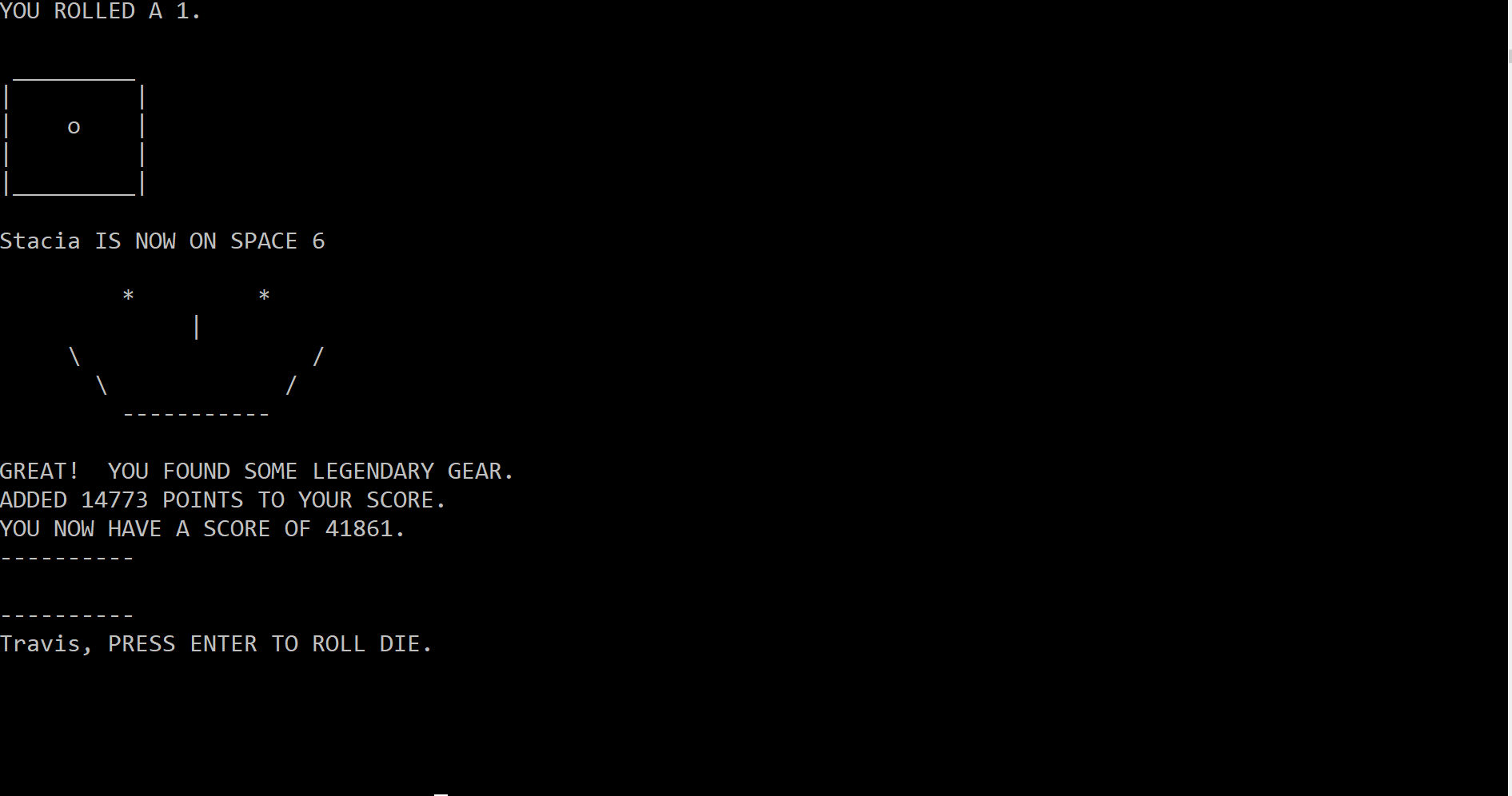
## SAMPLE OUTPUT OF PROGRAM RUNNING

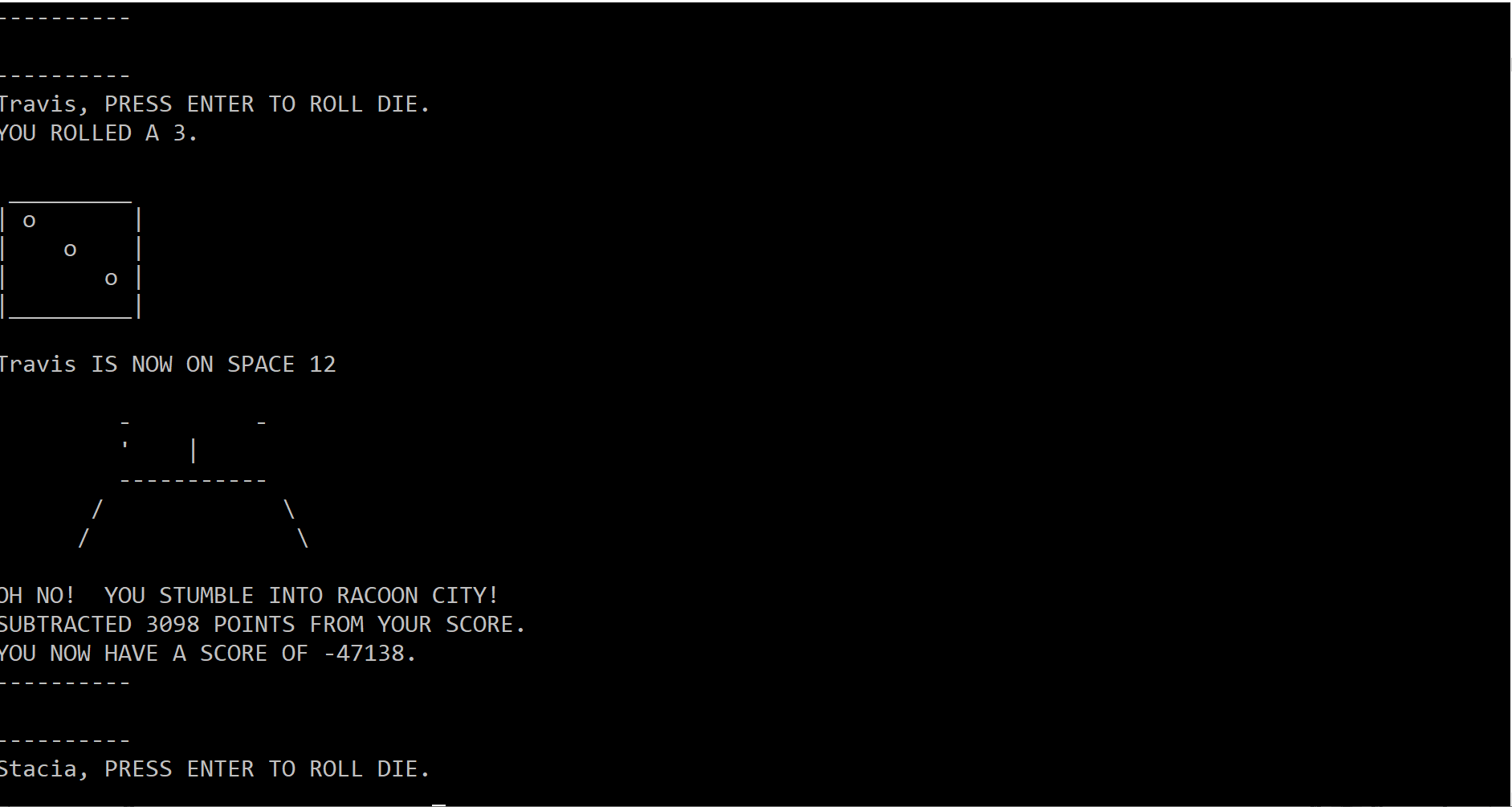
I have also provided a text sample output.

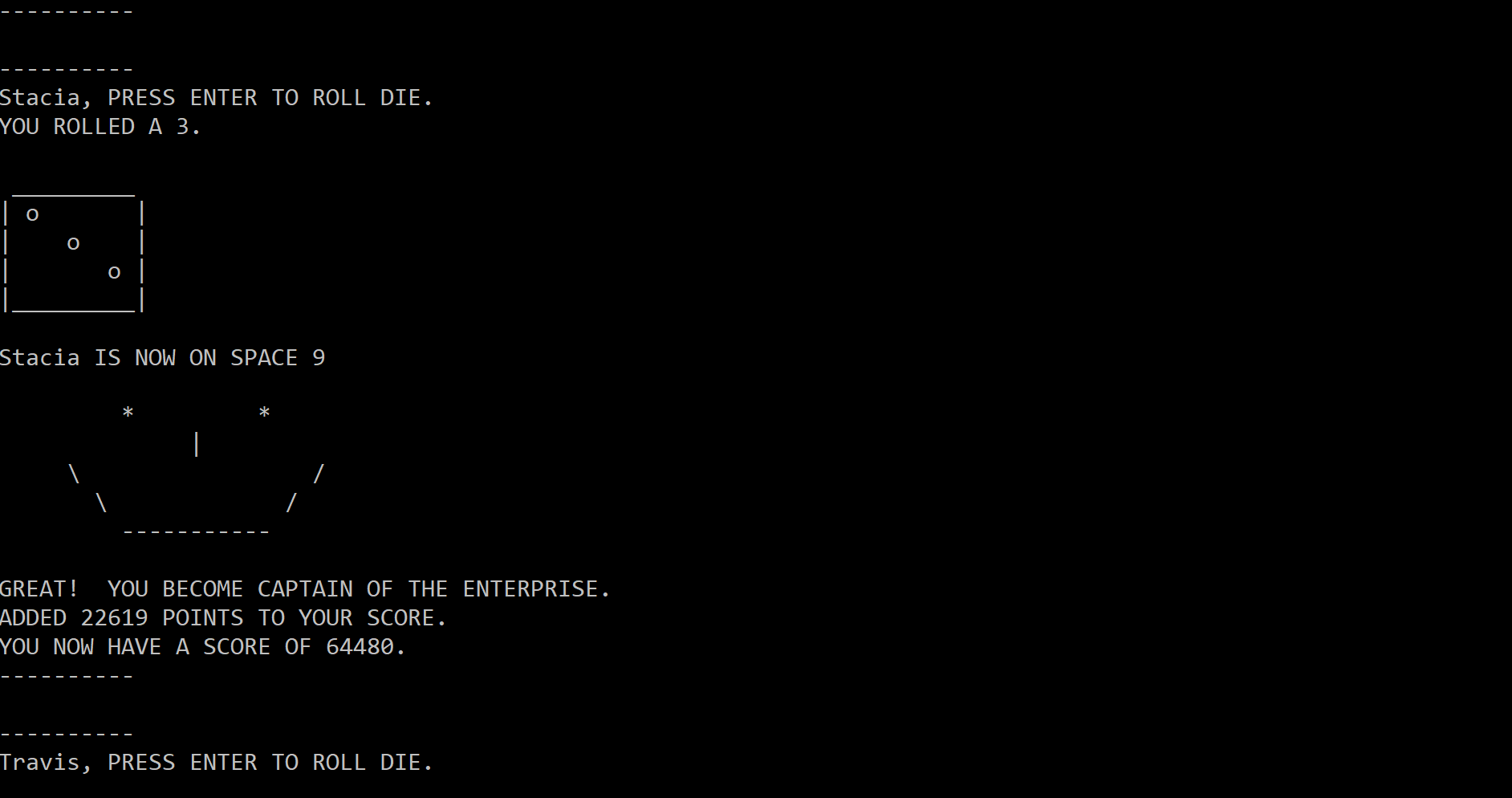


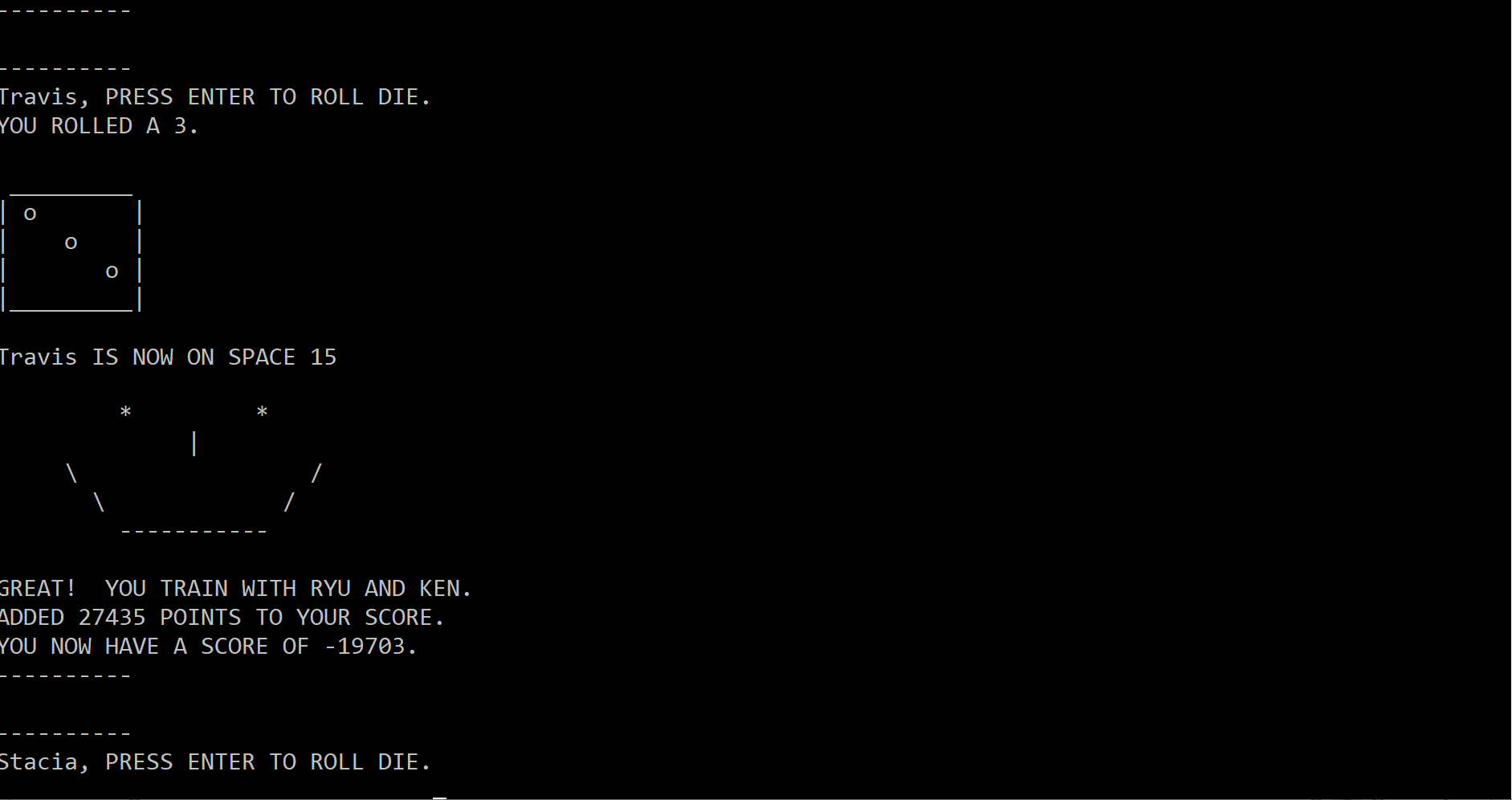


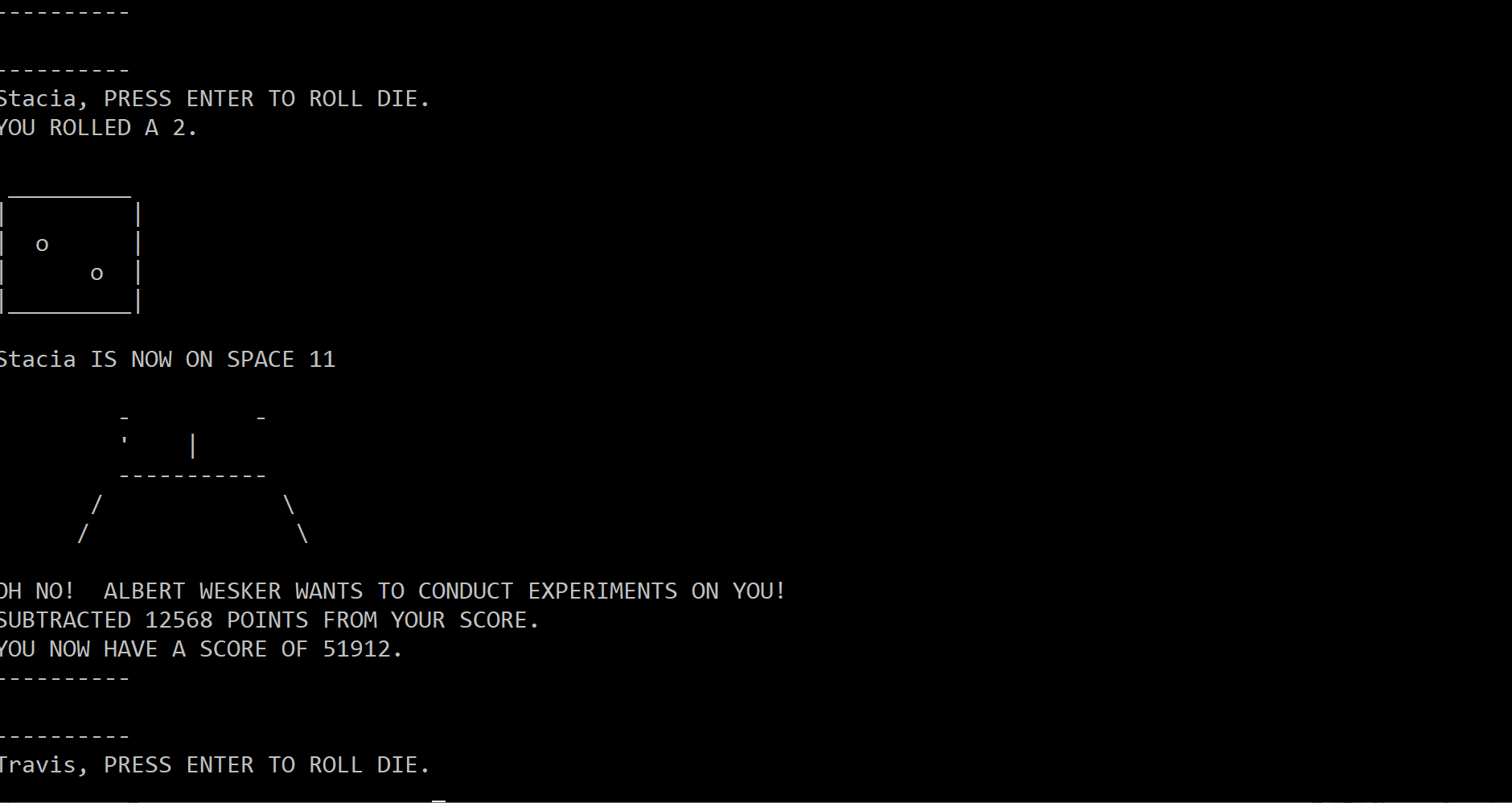
 

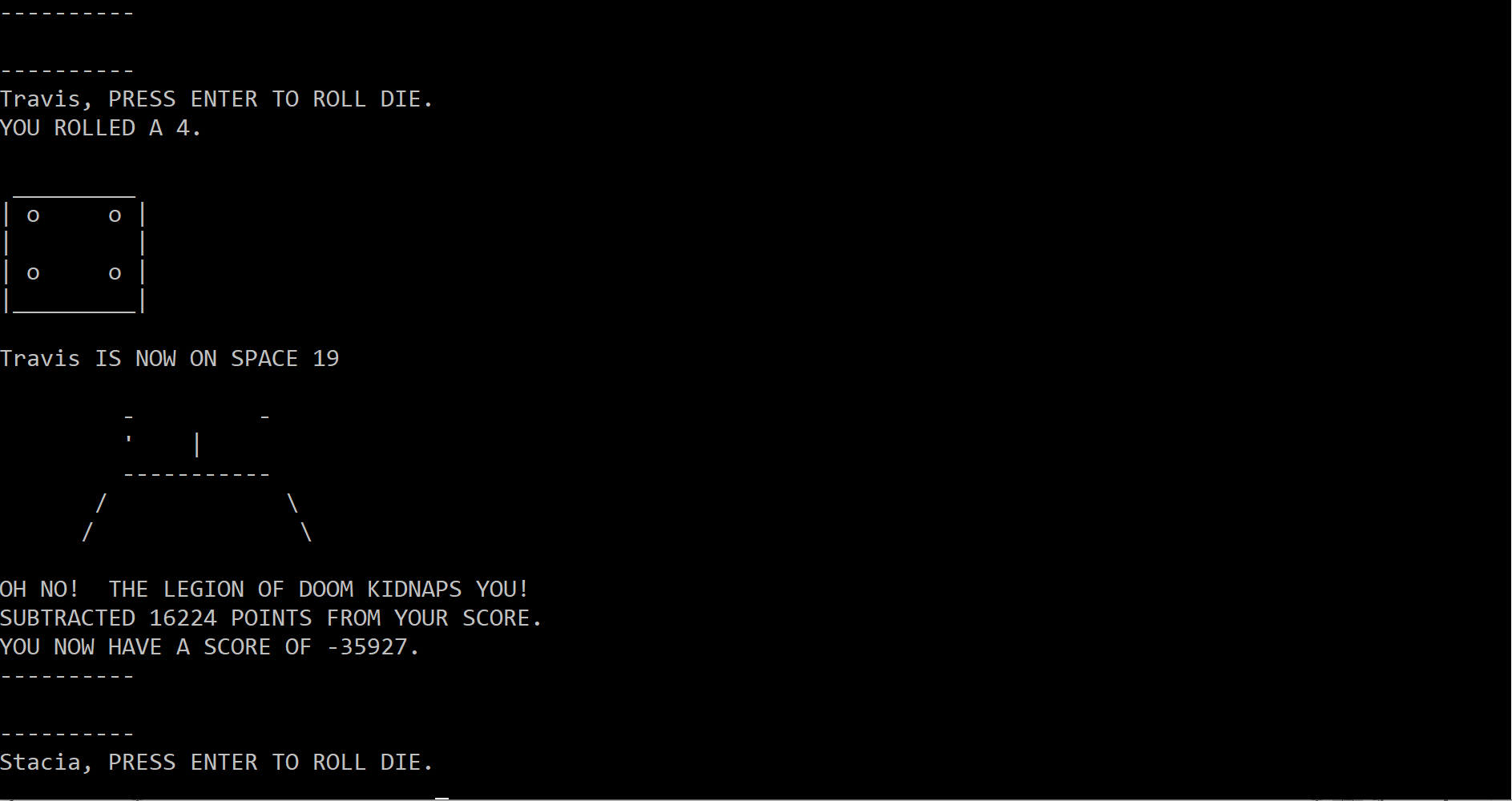


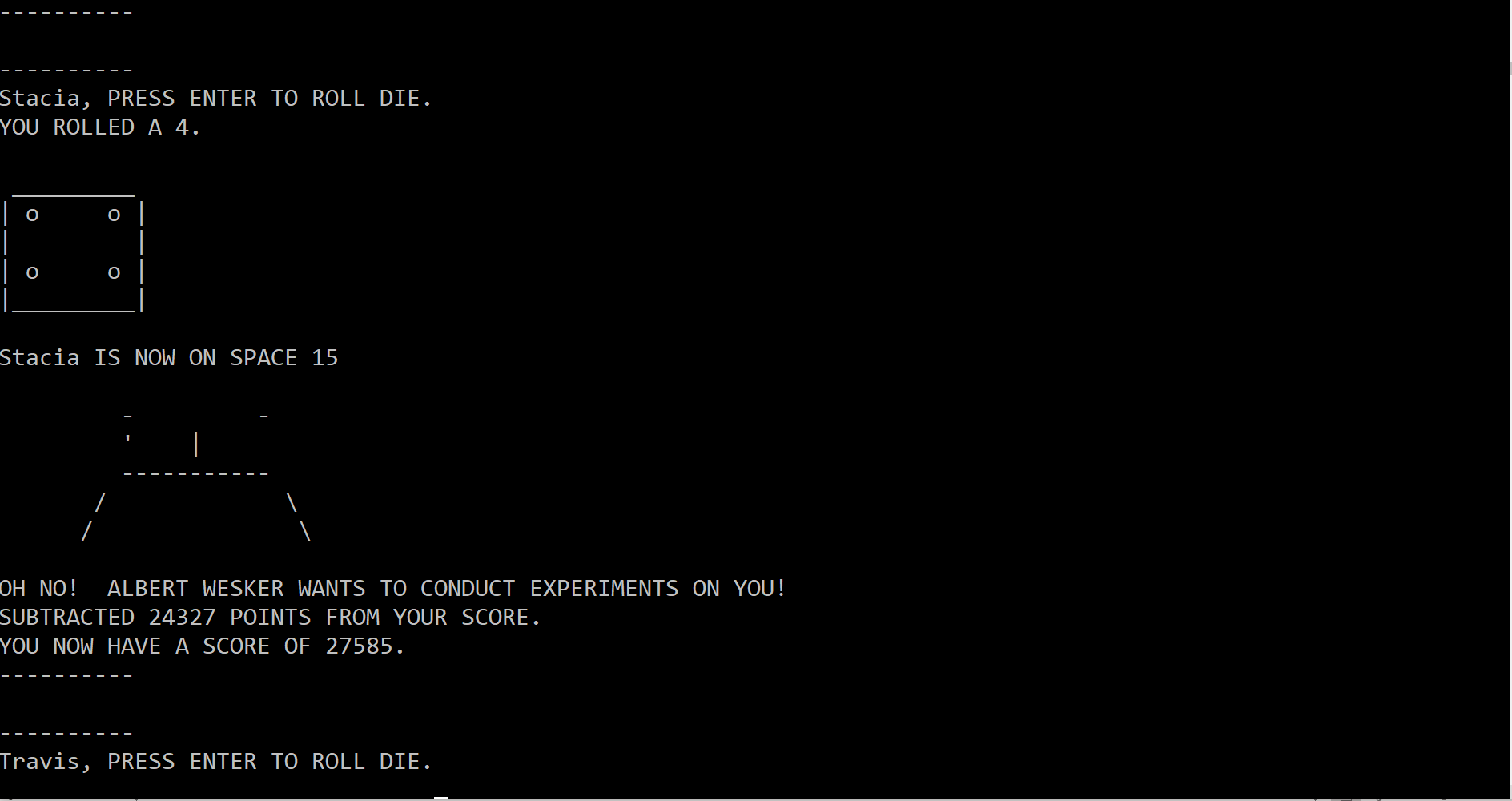


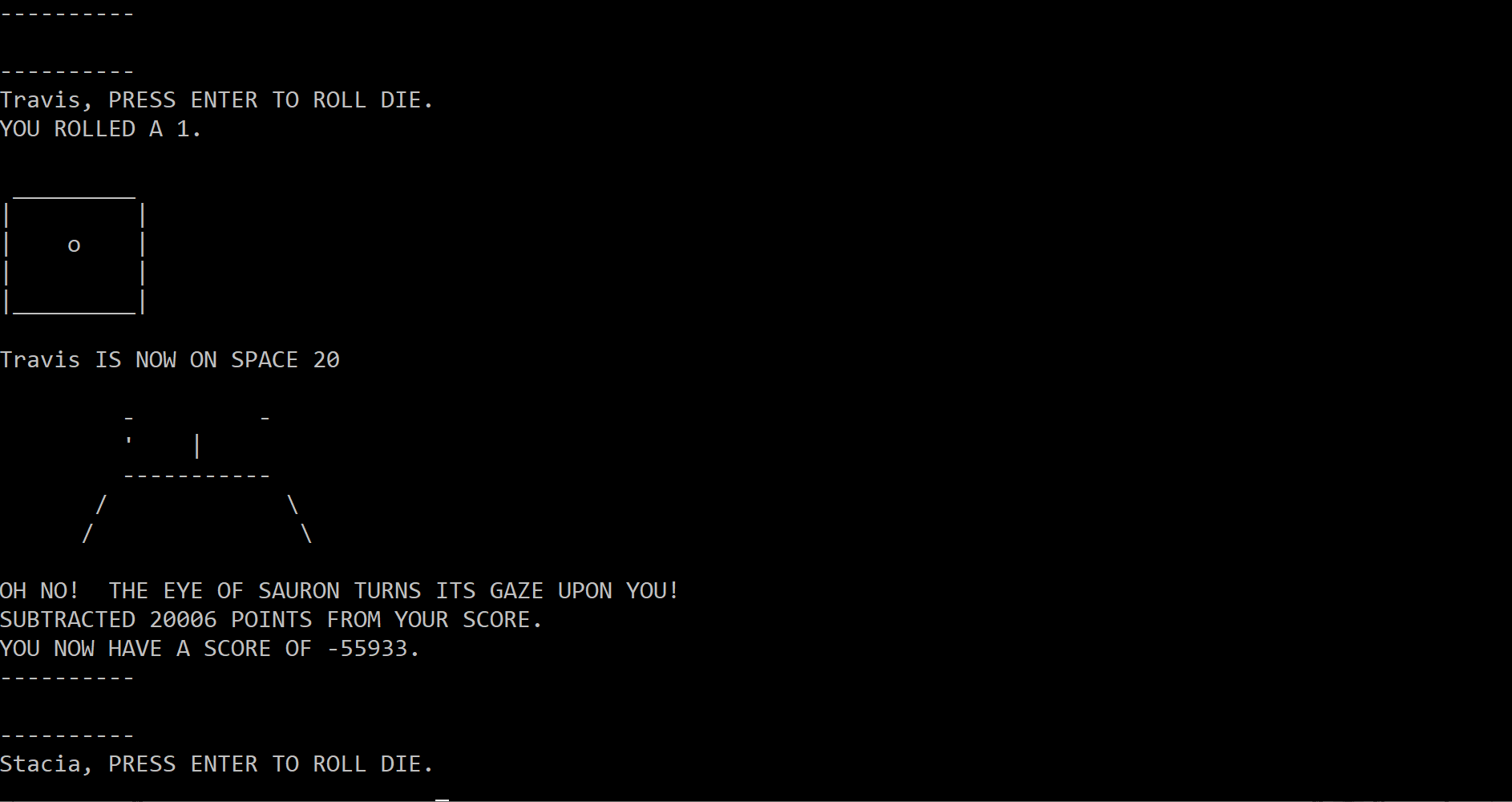


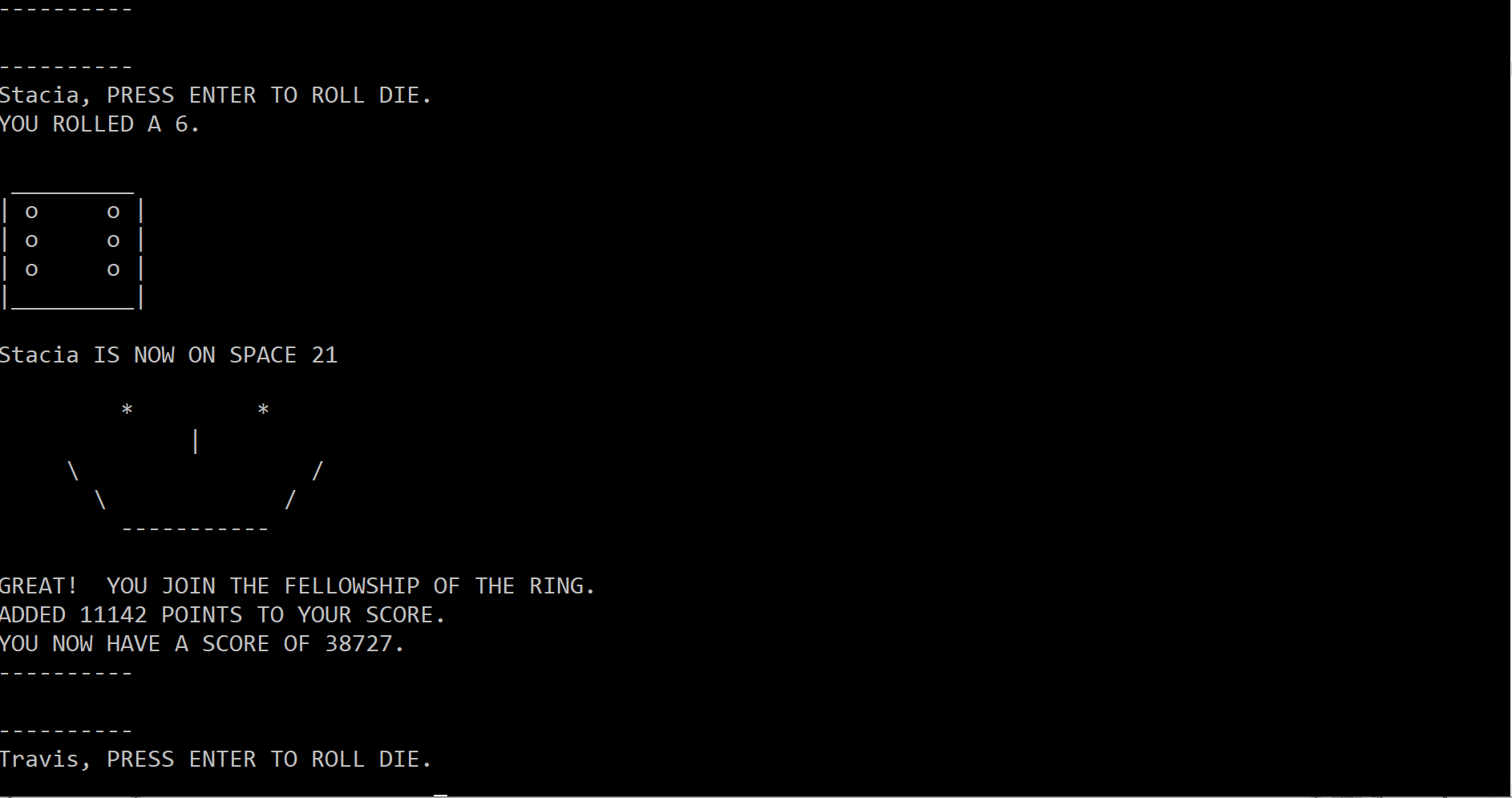


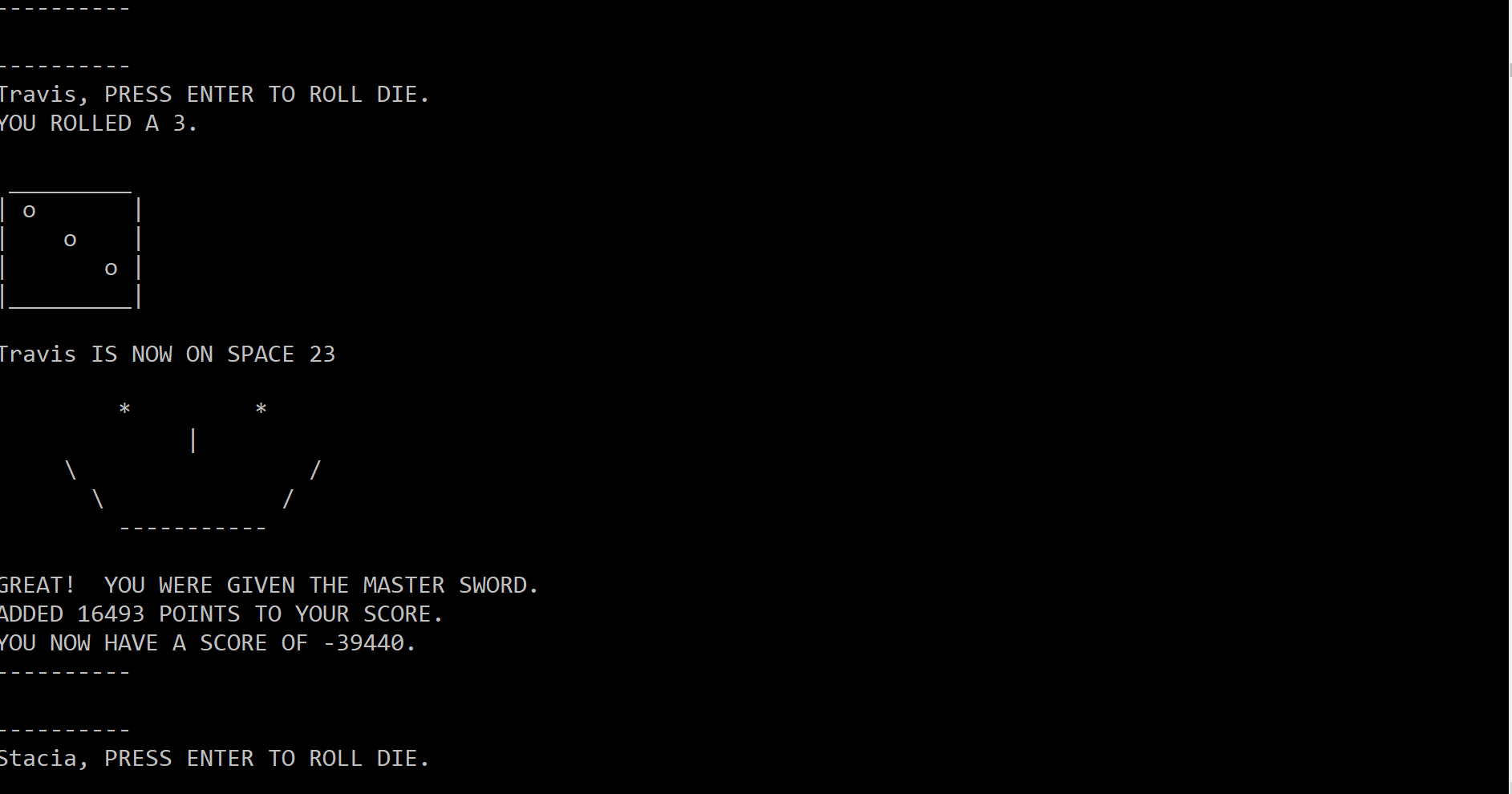


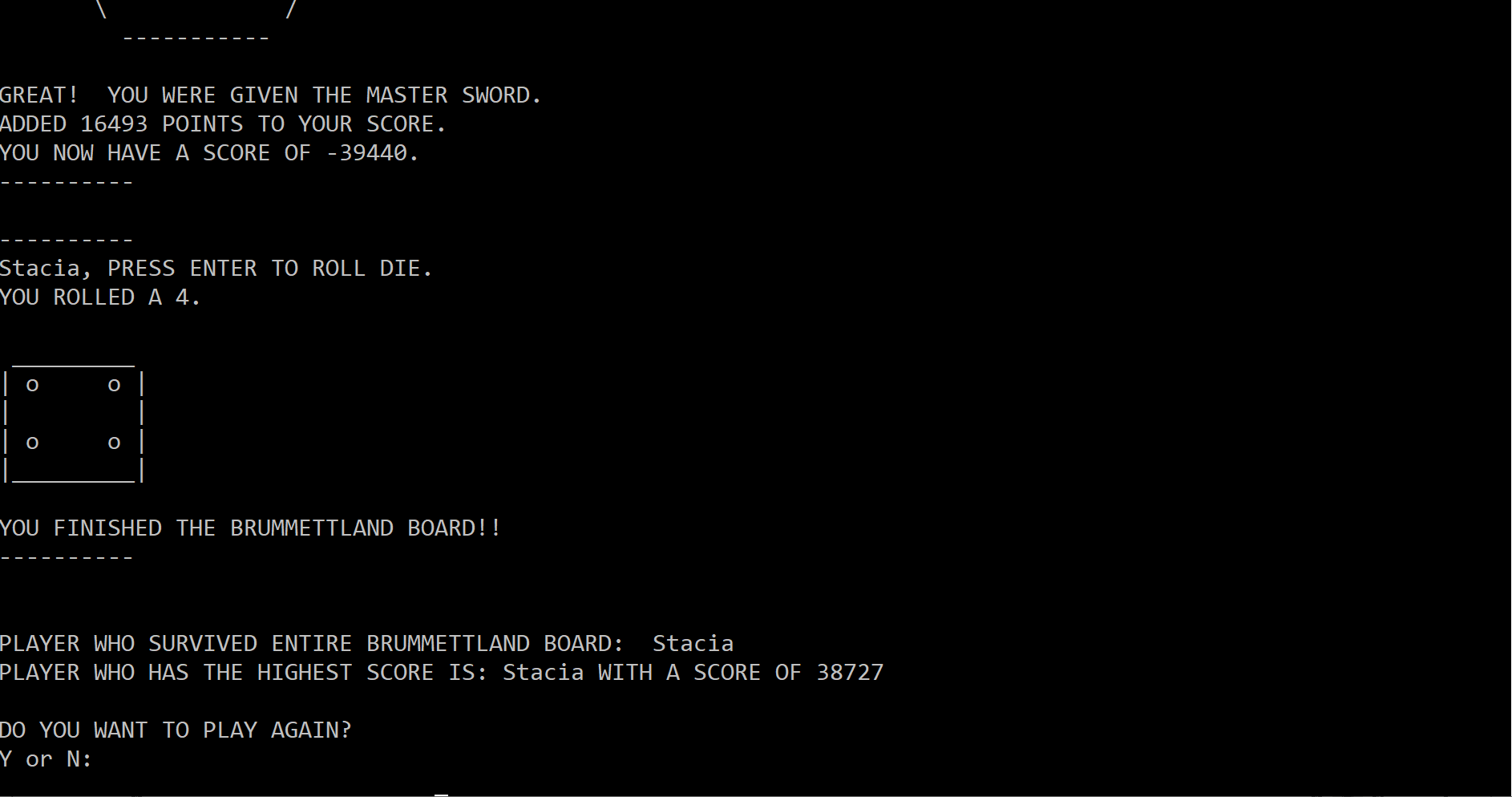












**CSC1300 / Program 3 grade sheet**

|  |  |
| --- | --- |
|  | **TOTAL (OUT OF 100)** |
| 15% compilation (does it compile?) | |
|  | **TOTAL (15 points)** |
|  | Program Compiles (15 points)  Program Does Not Compile (0 points) |
| 75% execution & SPECIFICATION  (Does it work? WAS PROBLEM SOLVED IN SPECIFIED WAY?) | |
|  | **TOTAL (no more than 75 points)** |
|  | **ARRAYS**   * Names array created correctly & used correctly throughout program **(5 points)** * boardSpace array created correctly & used correctly throughout program **(5 points)** * Score array created correctly & used correctly through program **(5 points)** |
|  | **NUMBER OF PLAYERS**   * allows up to 20, validates incorrect input **(5 points)** |
|  | **getPlayerNames() function**   - each player can enter their name **(5 points)** |
|  | **EACH PLAYER CAN TAKE A TURN**   * **rollDice() function** – function implemented as specified - player asked to press enter to roll, Die roll is random & ASCII art displayed  **(10 points)** * Board space is advanced same as die roll **(5 points)** * **activateActionOnSpace() function -** Good/ bad message based on die roll is extracted from file, and printed with ASCII Art happy/sad face, Points is random and either added/subtracted, file was closed after use **(15 points)** |
|  | **PlayerFinishedBoard() function** –  results were printed correctly - who survived entire board & who had highest score. **(15 points)** |
|  | **ALLOW USER TO PLAY AGAIN:** User can play again if they want to and the second time around it works like the first.  **(5 points)** |
|  | **OTHER: (points can also be deducted for the following problems)**  - code is efficient  - main function is defined at top of source file and programmer defined functions are under main   * constant variables were created for constants in the program |
| 10% documentation (is code readable & user-friendly?) | |
|  | **TOTAL (OUT OF 10)** |
|  | Comment block with title, author, date & purpose. Consistent indentions. |