Sneaky Series

**Introduction to Computer Programming**

Yashraj Kakkad  
AU1841036  
[kakkad.m@ahduni.edu.in](mailto:kakkad.m@ahduni.edu.in)  
+91-9898899179

Prayag Savsani  
AU1841035  
[prayag.s@ahduni.edu.in](mailto:prayag.s@ahduni.edu.in)  
+91-7990437837

# **Problem Definition**

# Goals:

1. **To crack any series problem of any competitive examination**
2. **To teach students how to crack series with hands-on problem solving**
3. **To learn from the user in case the program cannot crack the series**

# Problems solved:

* Series problems asked in big competitive exams are usually tough and most books covering this are not exhaustive. This project will be a **one stop destination** for anyone willing to master such problems. The “e-learning” section will teach him all the concepts from the basics while involving him.
* The program will be able to solve any **standard** (Arithmetic, Geometric, Prime Numbers, Fibonacci etc.) as well as many **complex series** involving relations with previous terms.
* The user will be able to identify his weak areas in his preparation to strategize his preparation accordingly.
* The user can practice a vast set of problems of which he can set the difficulty. The user will also be allowed to practice problems in a particular category only if he feels so.

# End users:

Competitive exam aspirants and program administrator.

# Highlighted features:

The program will basically have three main sections:

1. **E-learning:** Learn concepts, practice from ground zero in a particular topic, jump to a topic.
2. **Series cracker:** User enters the first four terms of any series. The program calculates the next three terms and explains the logic also. In case the program cannot crack, the user will be allowed to convey this message to the administrator.
3. **Quiz section:** The user can practice problems, identify his weak areas and get statistical data about his performance.

The administrator can:

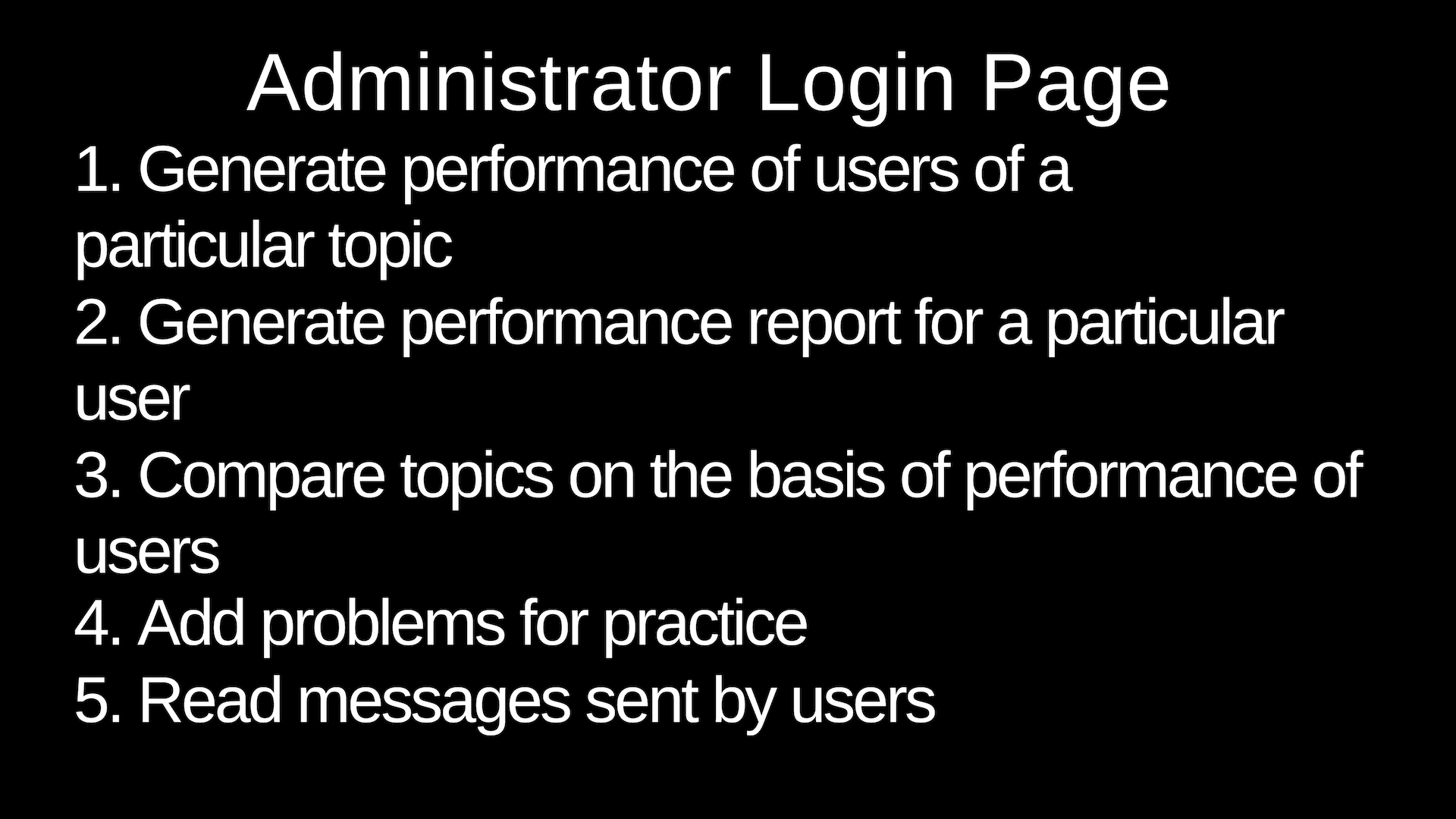
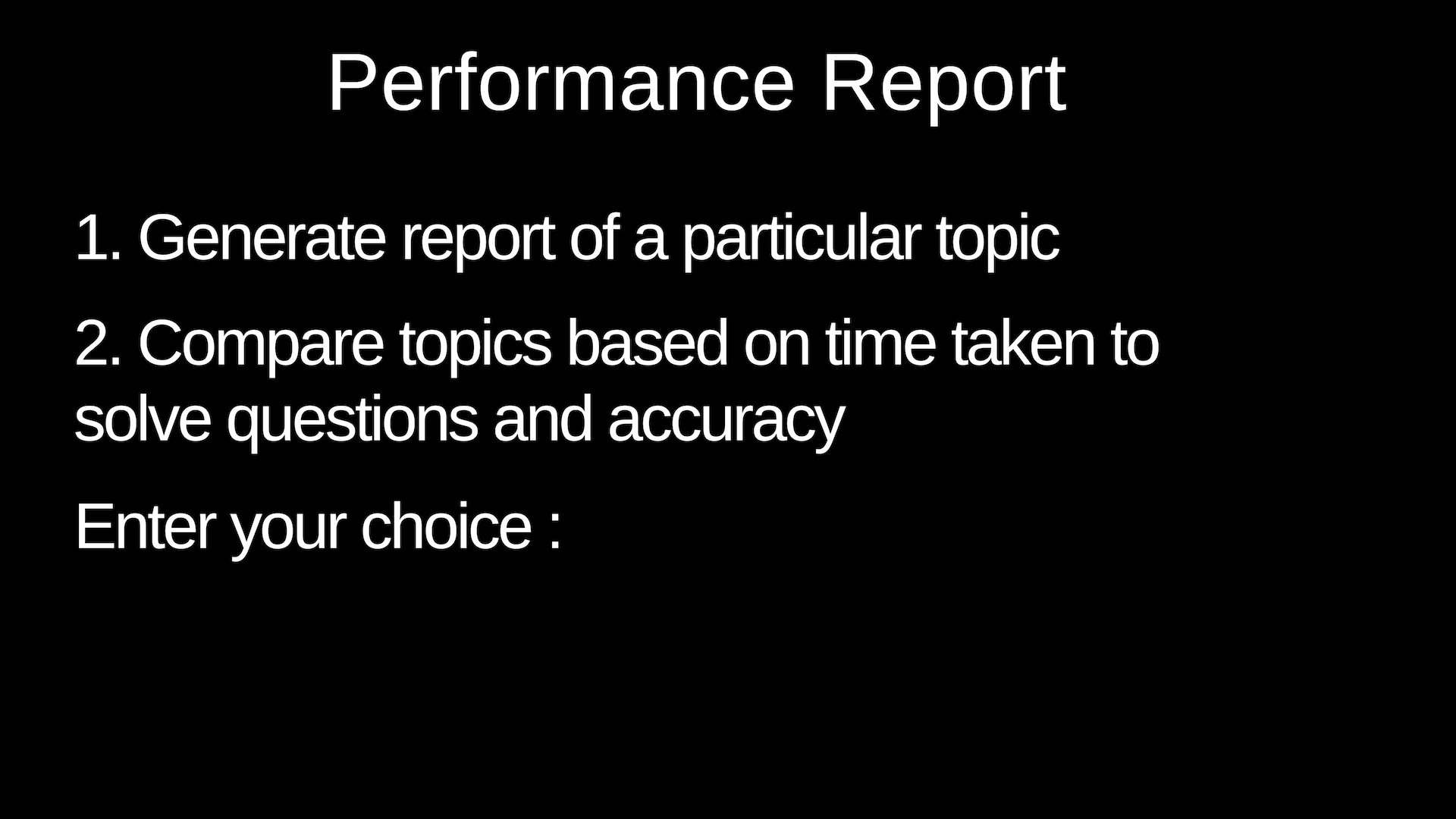
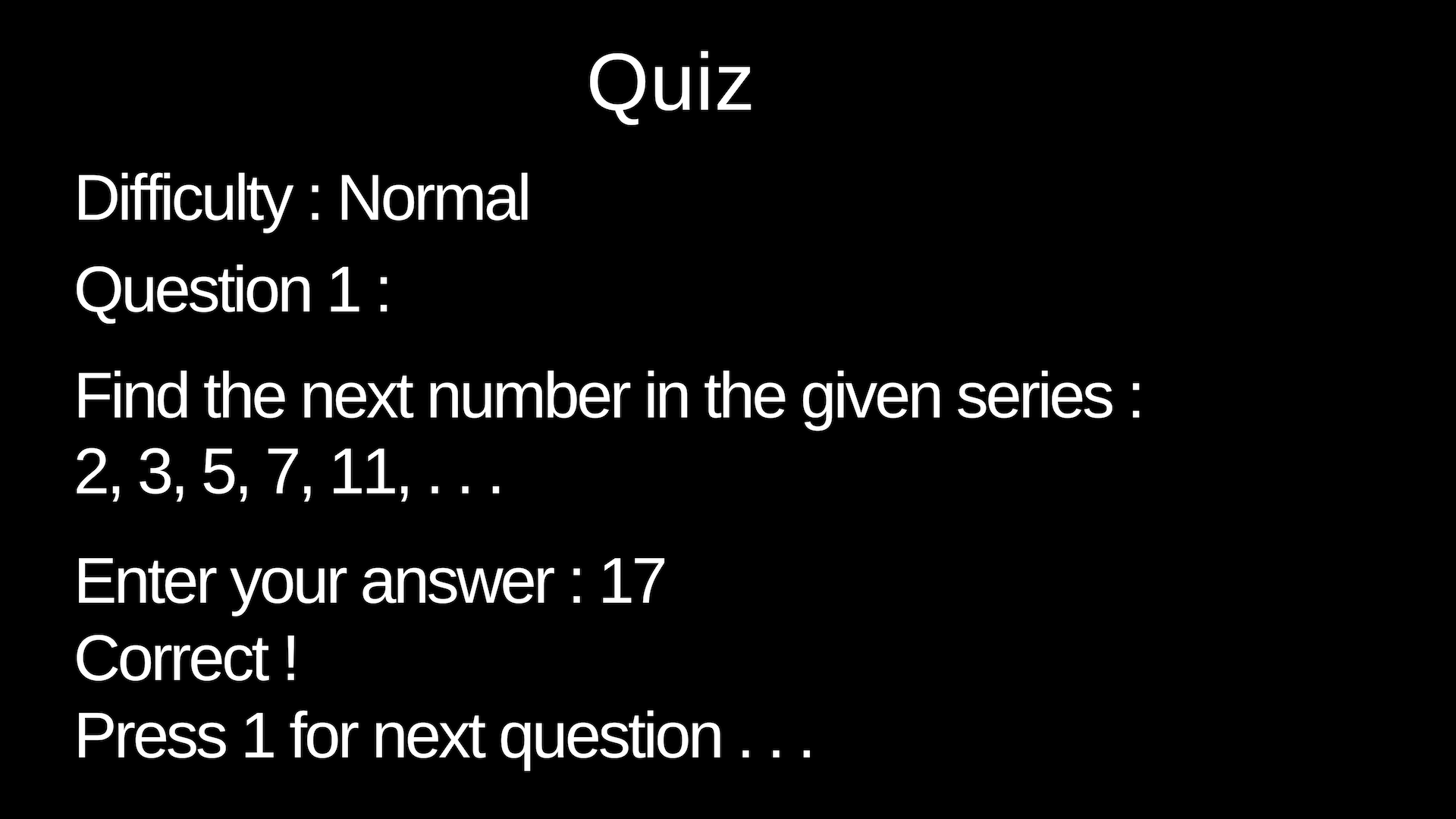
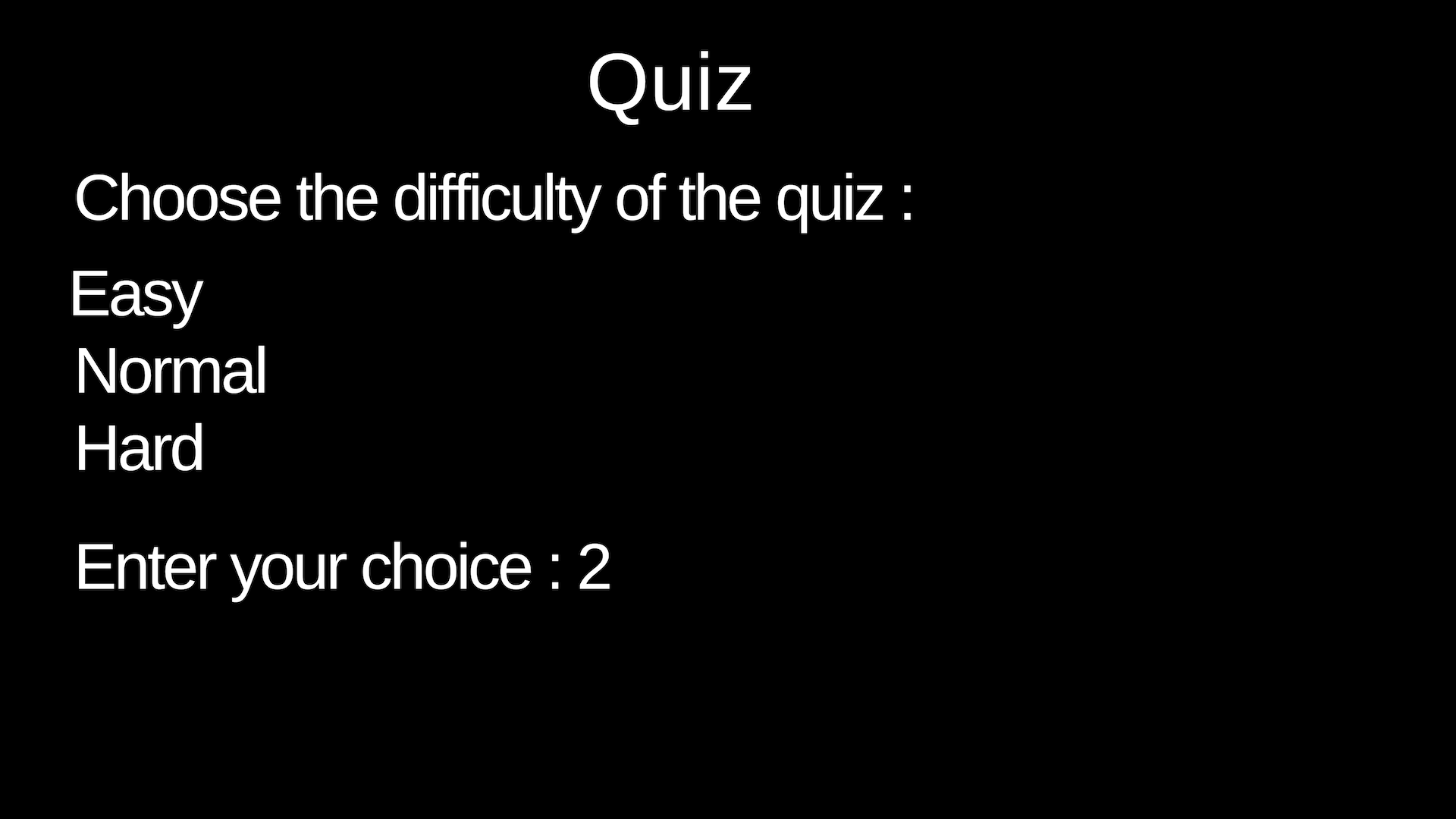
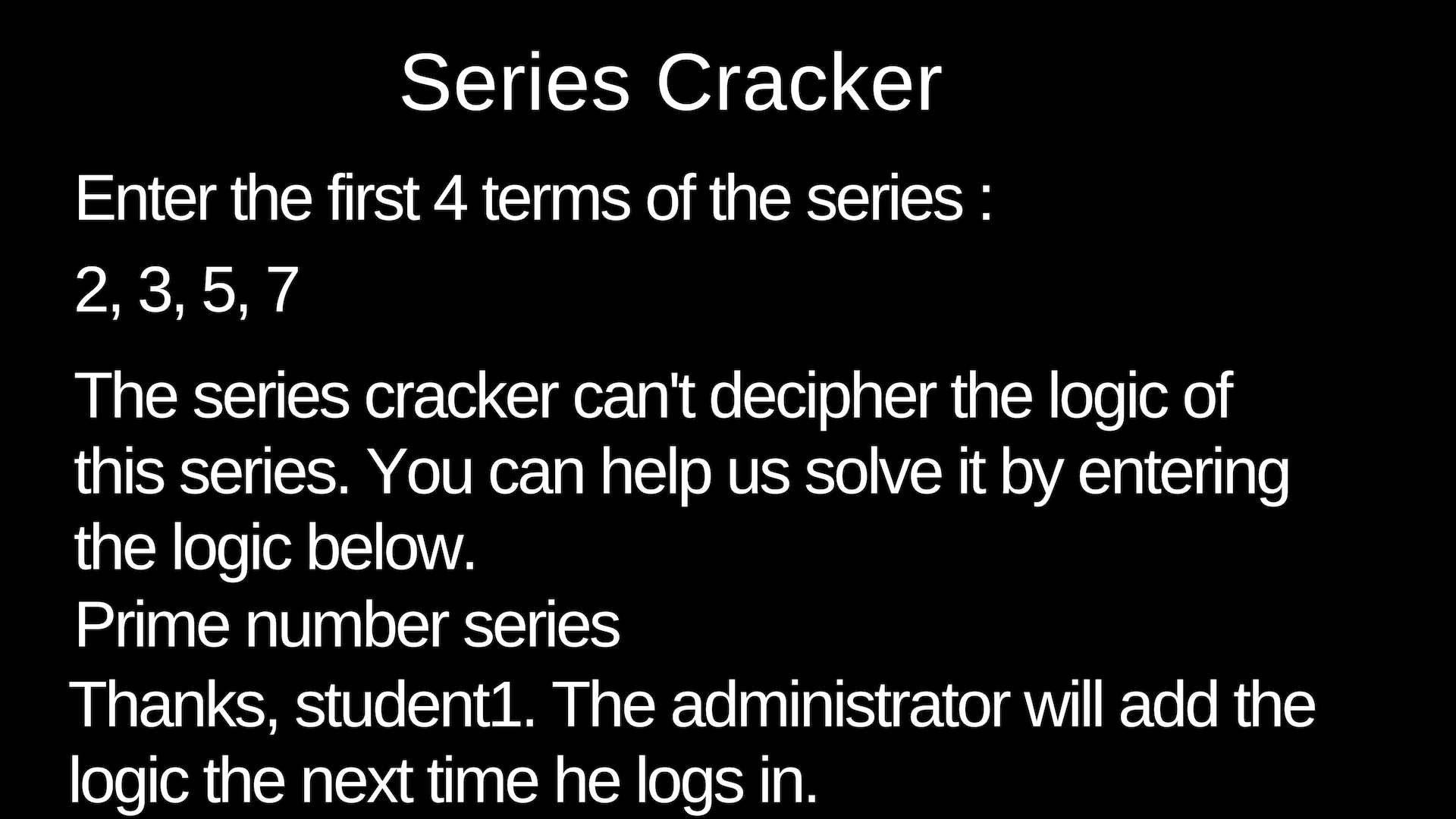
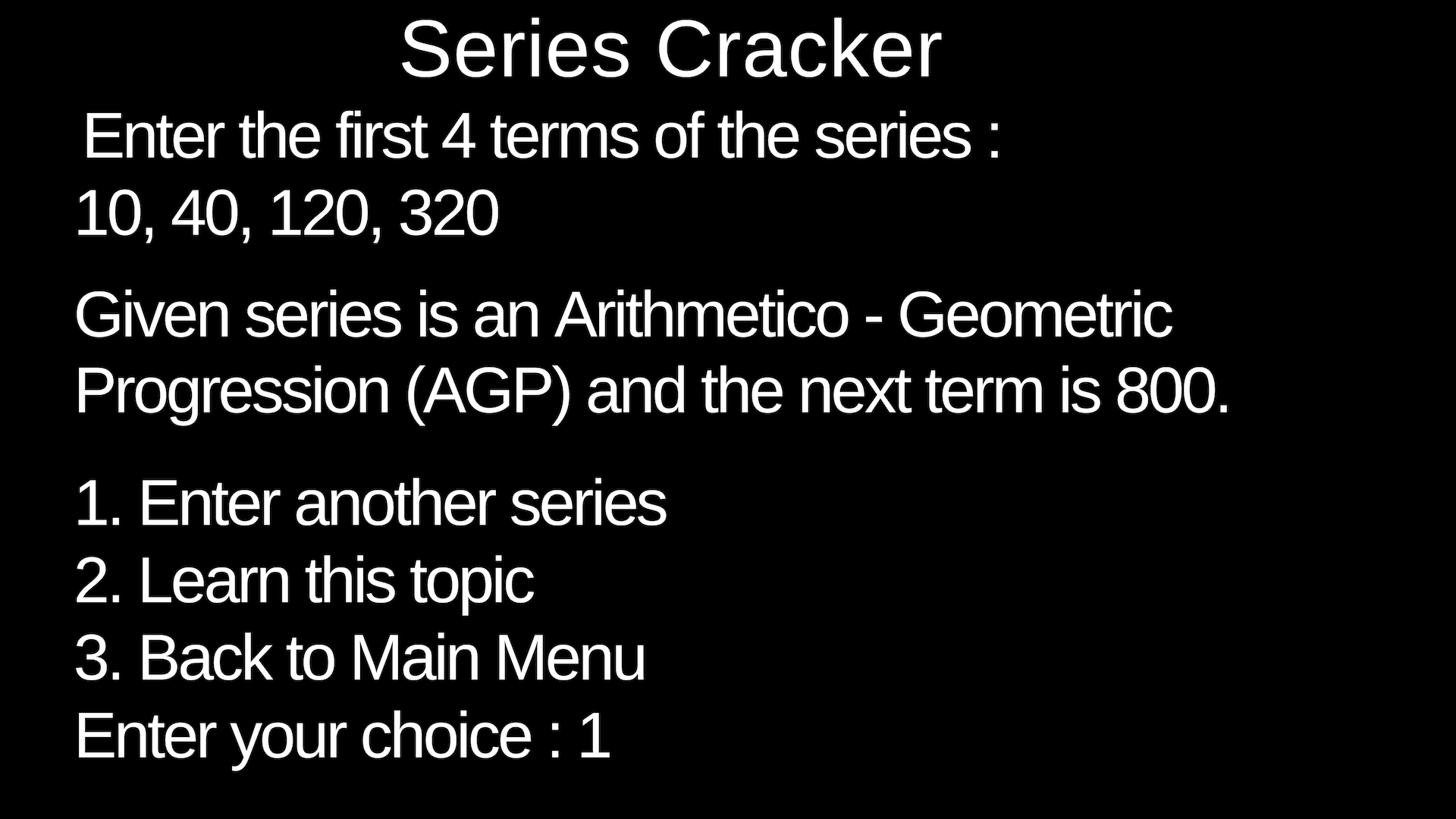
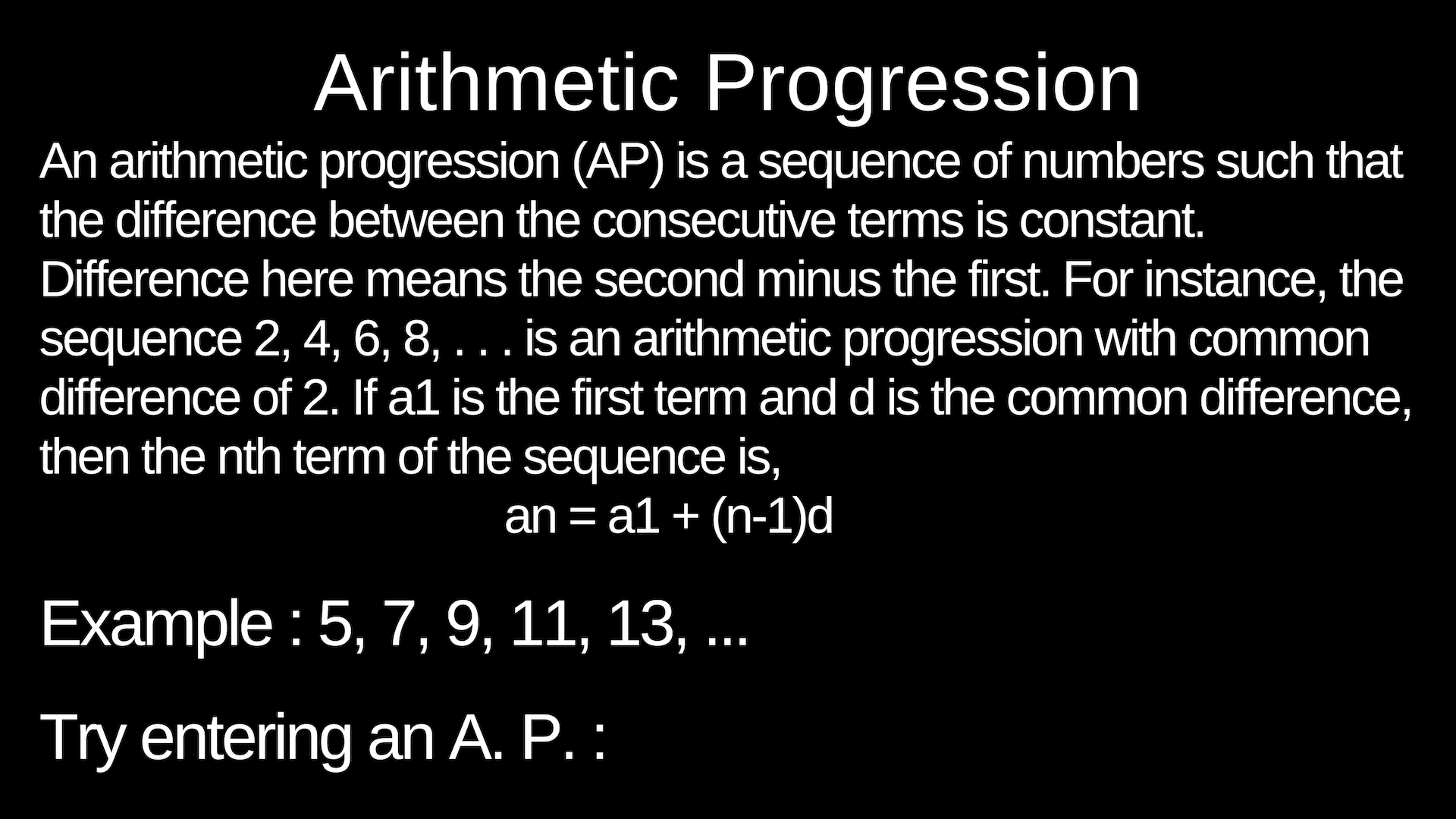
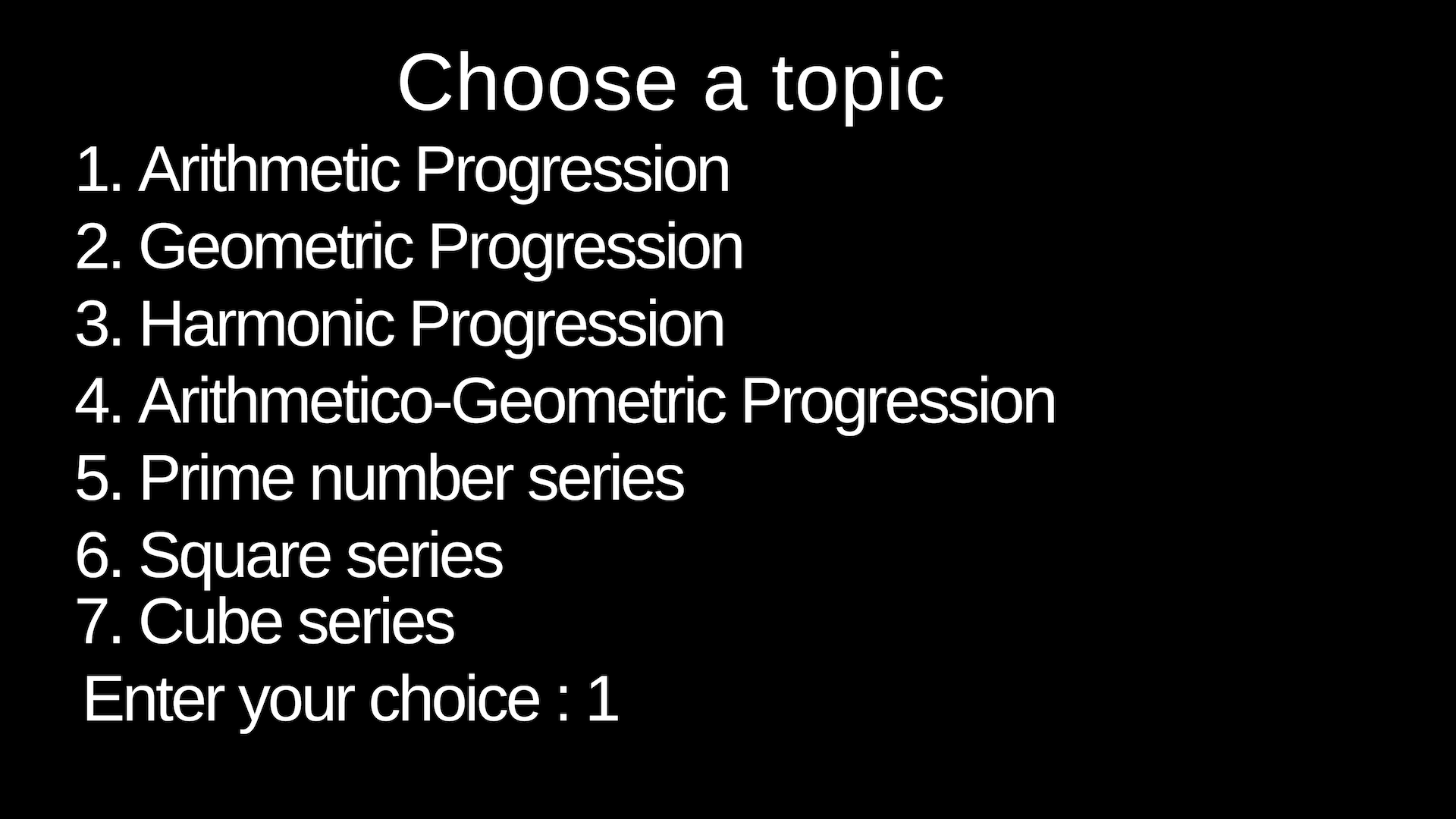
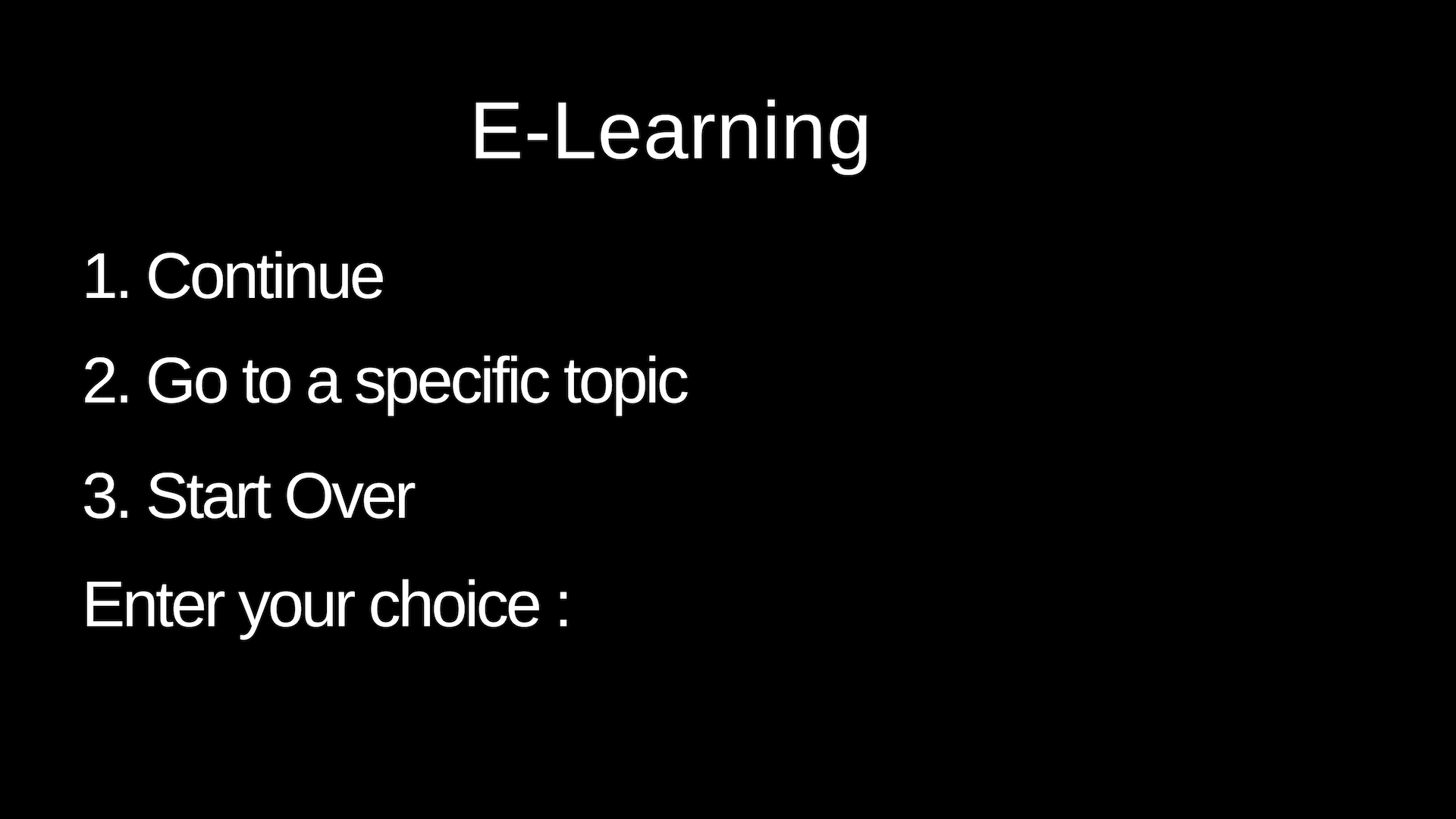
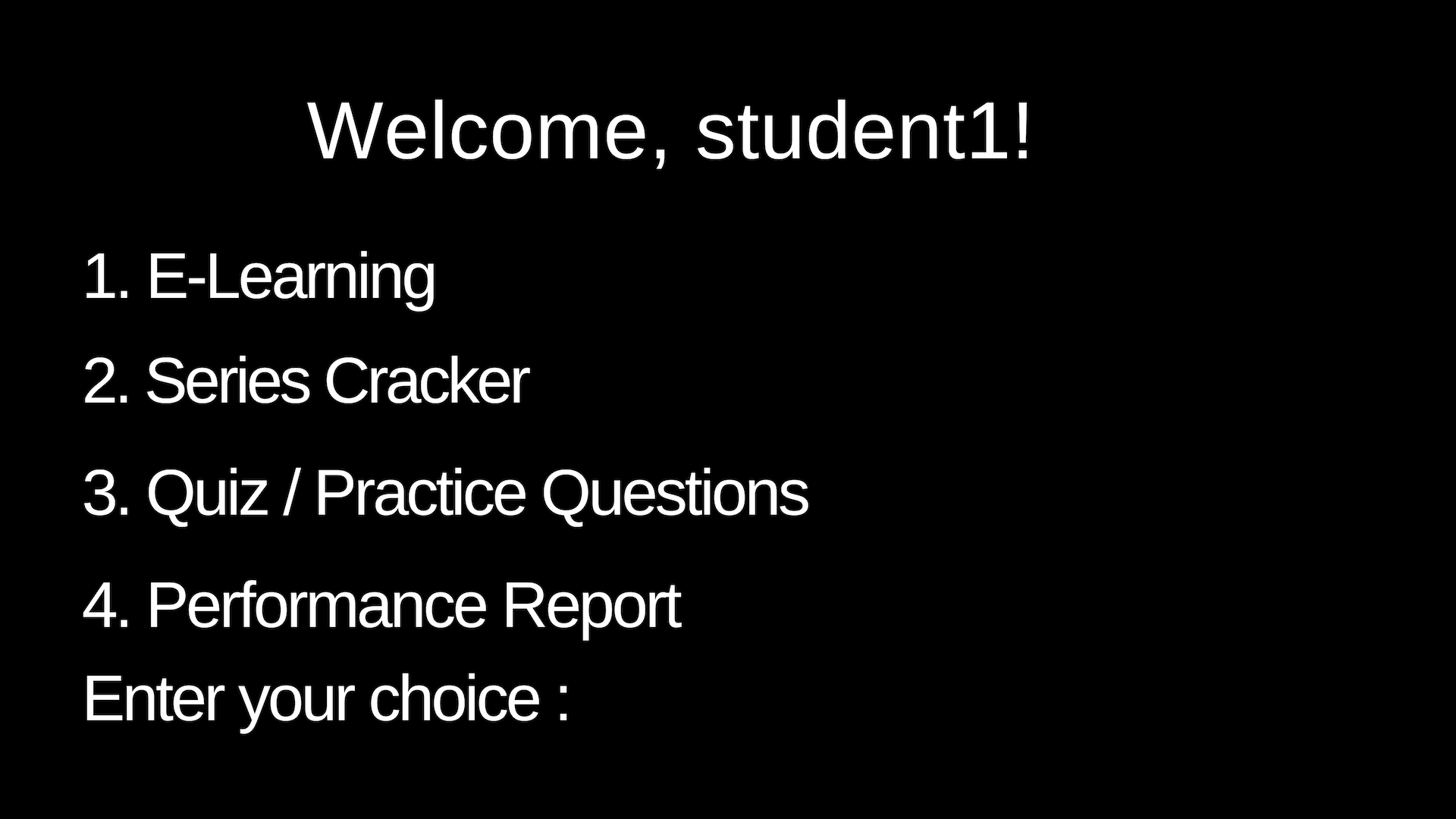
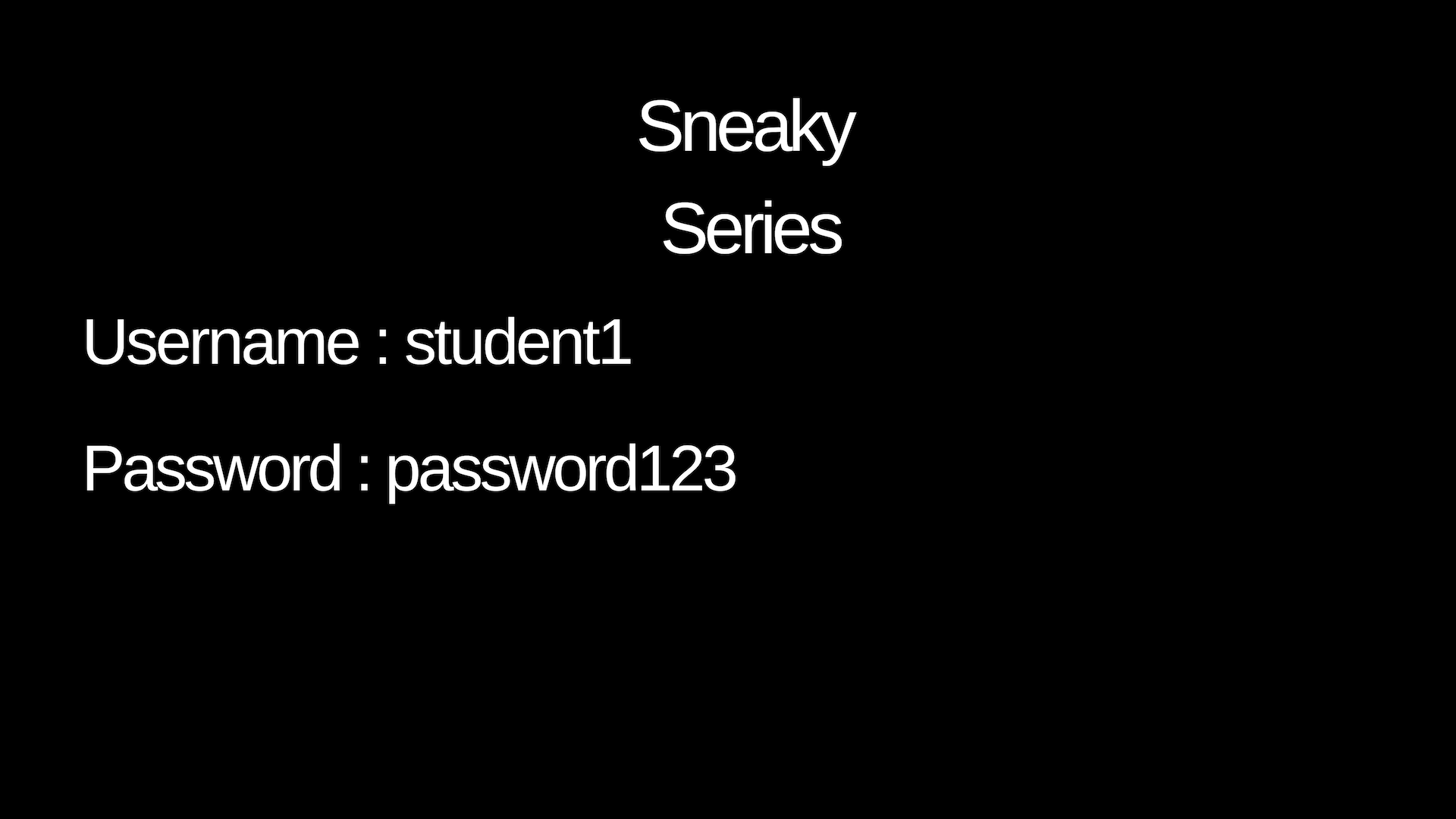
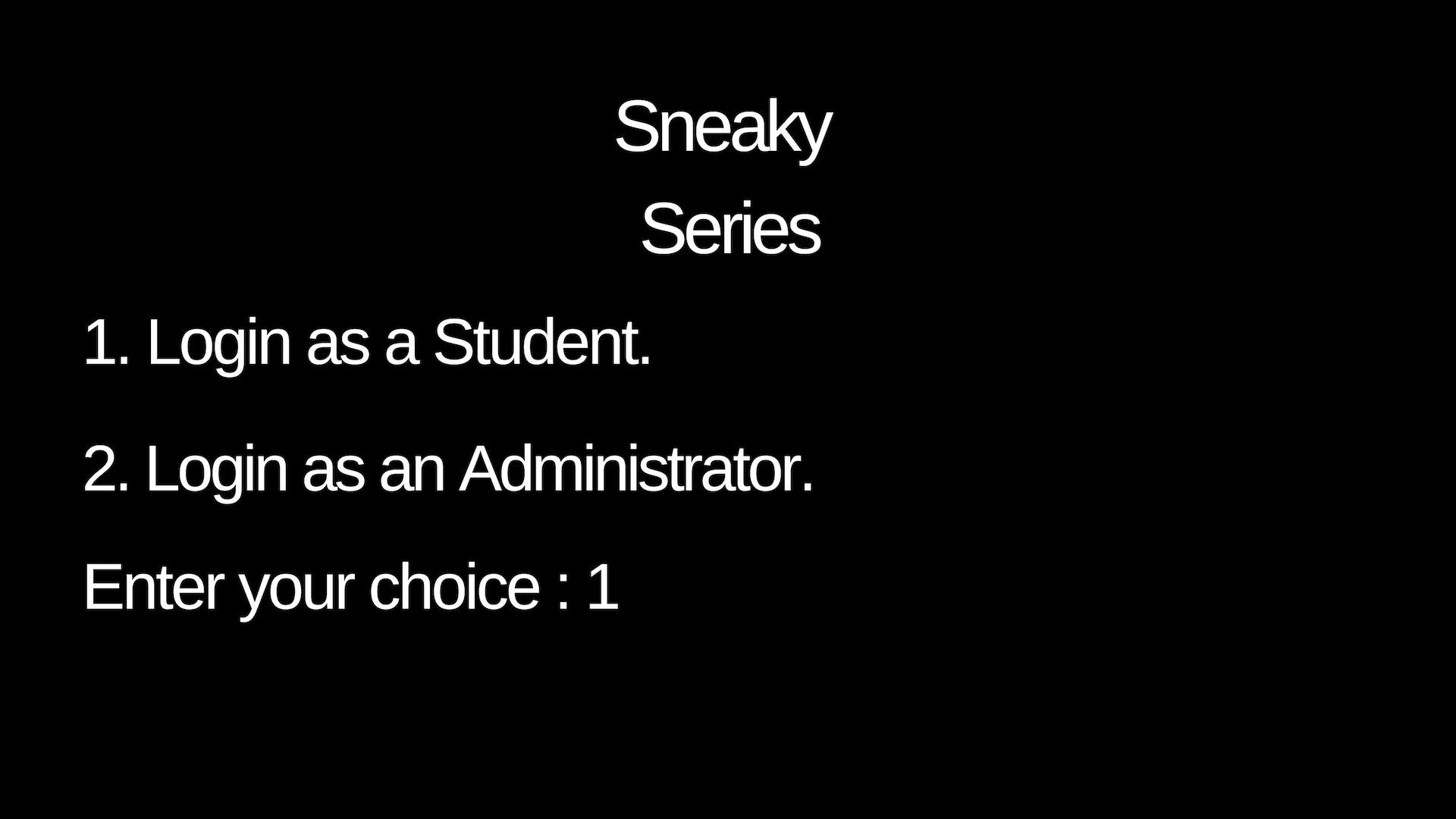
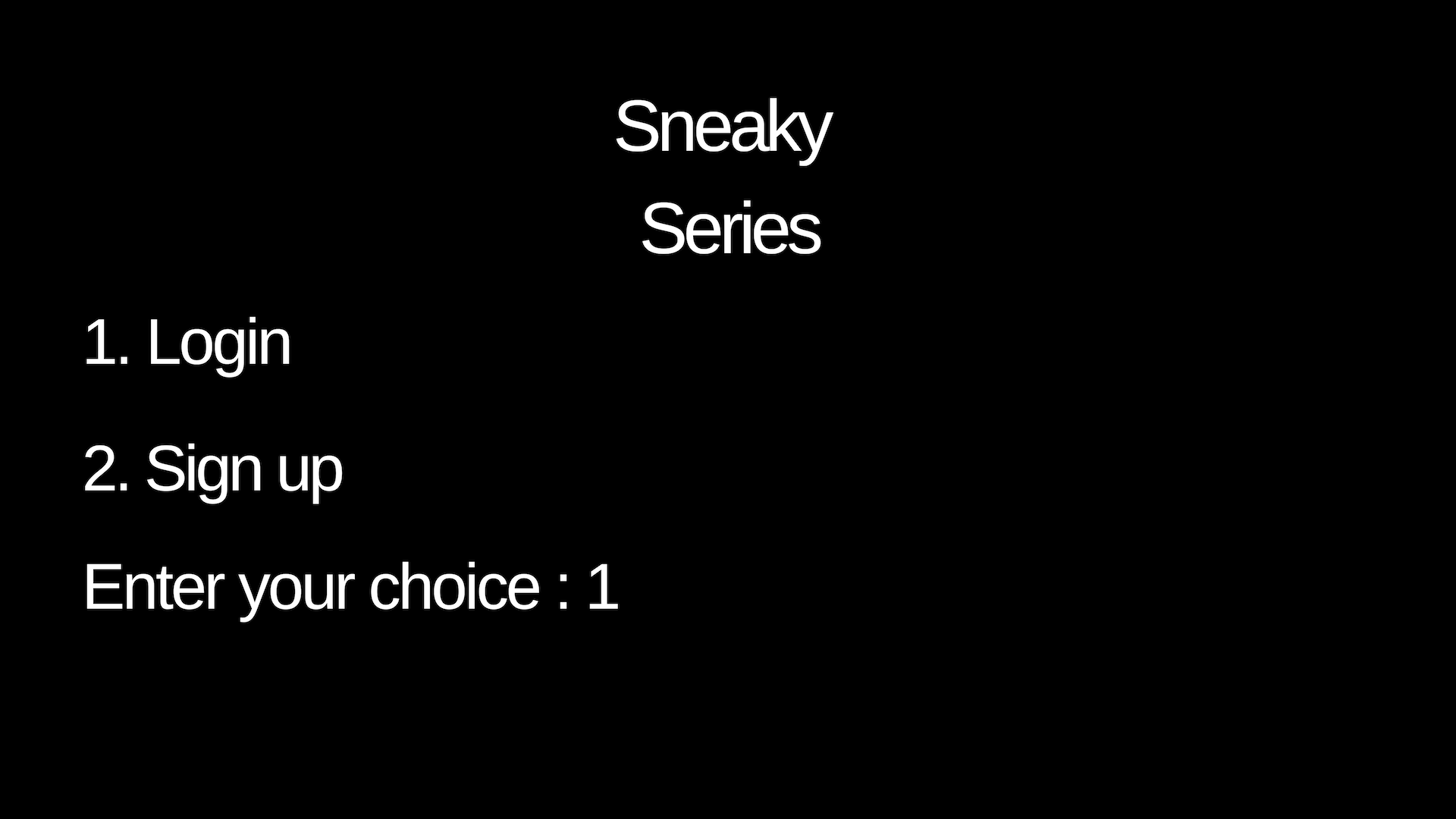
1. Generate data for any particular user
2. Generate data for any particular topic
3. Identify the easiest and toughest problems in general based on accuracy and time spent of users.
4. Read the inputs from users about the series problems not solved by the program.

# Important system outputs:

Average time spent for any question, weak topics, strong topics - for one student as well as for all students. List of easiest and toughest problems based on accuracy and time spent.

# **Project Design**

# User Interface Design



# 

# Database Design

1. **E-learning content files**

* “progressions.txt”
* “advancedprogressions.txt”
* “multipleseries.txt”
* “previousterms.txt”
* “squaresandcubes.txt”
* “standardseries.txt”

**2. Learning progress** (learningprogress.csv)

**0** - not completed  
 **1** - completed

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User** | Progressions | Advanced Progressions | Multiple Series | Previous Terms | Squares and Cubes | Standard Series |
| userid1 | 1 | 1 | 1 | 0 | 0 | 0 |
| userid2 | 1 | 1 | 0 | 0 | 0 | 0 |
| userid3 | 1 | 1 | 1 | 1 | 1 | 1 |
| userid4 | 1 | 1 | 1 | 0 | 0 | 0 |

**3.** **Quiz database** (quizdatabase.csv)

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No | Field name | Data type | Sample data |
| 1. | Question Code | Integer | 132 |
| 2. | Difficulty  0 - easy  1 - medium  2 - difficult | Integer | 0 |
| 3. | Term 1 | Float | 3.75 |
| 4. | Term 2 | Float | 4.25 |
| 5. | Term 3 | Float | 4.75 |
| 6. | Term 4 | Float | 5.25 |
| 7. | Term 5 (Answer) | Float | 5.75 |
| 8. | Term 6 (Answer) | Float | 6.25 |

**4. Student score tally**

**-1** - not attempted  
 **0** - wrong answer  
 **1** - correct answer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | #QCODE1 | #QCODE2 | #QCODE3 | #QCODE4 | Average time per question (in last quiz) |
| userid1 | -1 | 1 | 0 | 1 | 30 |
| userid2 | 1 | 0 | 1 | -1 | 72 |
| userid3 | 1 | 1 | -1 | 0 | 41 |

**5. User messages** (messages.csv)

|  |  |
| --- | --- |
| Username | Message |
| userid1 | Thank you for making this wonderful program. It made my life easy! |
| userid2 | Please add successive terms of exponential function as a series. Thank you |

**6.** **Questions entered by users in series cracker** (so that admin can introduce newer questions in his quiz for other users) (userquestions.csv)

# Program Design

**Structure: Quiz Problem**  
Members: Question Code (int), Difficulty index(int), four terms for question(float), two terms for answer(float)

**Functions along with their prototypes :**  
void menu();  
void learn();  
void quiz();  
void seriescracker();  
void userperformance();  
void addquiz();  
//E-learning functions  
void learnhome(int section);  
void Lprogressions();  
void Ladvancedprogressions();  
void Lstandardseries();  
void Lsquaresandcubes();  
void Lpreviousterms();  
void Lmultipleseries();  
void Lnavigation(int section);  
//Series cracker functions  
void a\_progression(float w, float x, float y, float z, float\* t1, float\* t2);  
void g\_progression(float w, float x, float y, float z, float\* t1, float\* t2);  
void h\_progression(float w, float x, float y, float z, float\* t1, float\* t2);  
void prime(float w, float x, float y, float z, float\* t1, float\* t2);  
void da\_progression(float w, float x, float y, float z, float\* t1, float\* t2);  
void fibonacci(float w, float x, float y, float z, float\* t1, float\* t2);  
void sumrelation(float w, float x, float y, float z, float\* t1, float\* t2);  
void sumproduct(float w, float x, float y, float z, float\* t1, float\* t2);  
void square\_series(float w, float x, float y, float z, float\* t1, float\* t2);  
void cube\_series(float w, float x, float y, float z, float\* t1, float\* t2);  
void factorial\_series(float w, float x, float y, float z, float\* t1, float\* t2);  
void squaresum\_series(float w, float x, float y, float z, float\* t1, float\* t2);  
void productpretwo\_series(float w, float ;x, float y, float z, float\* t1, float\* t2);  
float factorial(float x);

//Other functions

void loginmodule();  
void userreport();

//Admin functions  
void readmessages();  
void overallreport();  
void userreportadmin();  
void comparetopics();

# 

# 

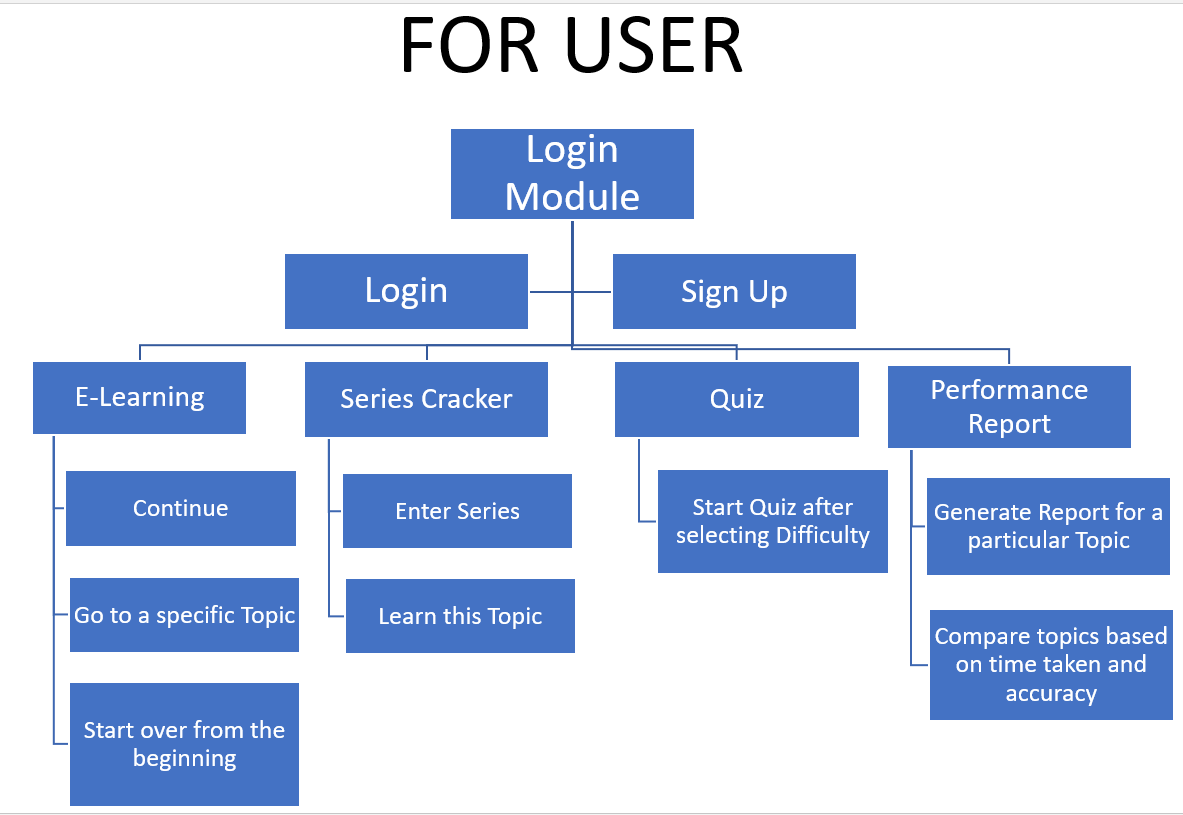
# 

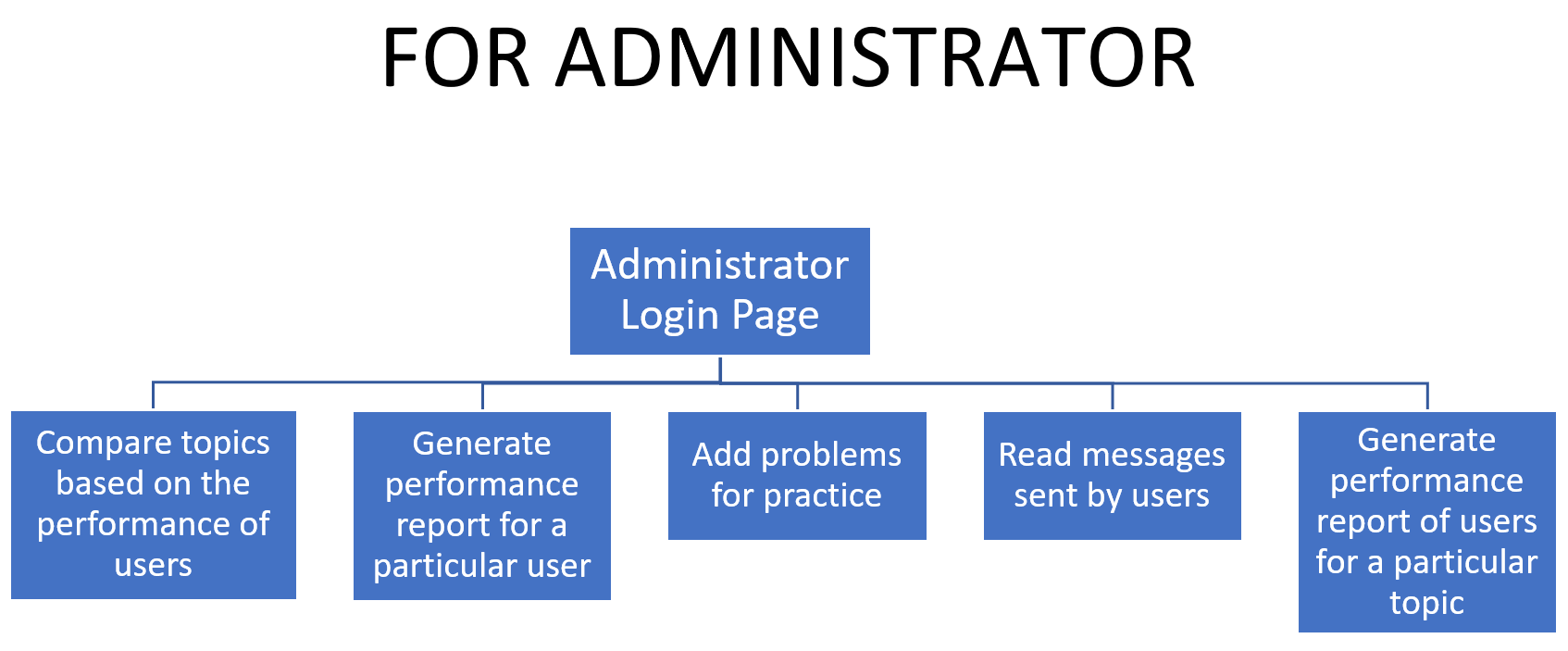
# 

# 

# 

# Structure Chart





# Important algorithms used in the project

**Series Cracker :**

1. Accept the four terms from the user.
2. Check if it is an arithmetic progression
3. If false, check for geometric progression
4. If false, check for harmonic progression
5. If false, check if they are consecutive prime numbers
6. If false, check if the differences are in arithmetic progression
7. If false, check if the series is of the form x(n) = k\*x(n-1)+j
8. If false, check if the series is square of consecutive terms
9. If false, check if the series is cube of consecutive terms
10. If false, check if the series is factorial of consecutive terms
11. If false, check if the series is of the form x(n) = k\*x(n-1)^2+j
12. If false, check if the series is of the form x(n)=k\*x(n-1)\*x(n-2)
13. If false, check if the series is of the form x(n) = k\*x(n-1) + j\*x(n-2)
14. If false, ask the user if he wants to send this question to admin

**Checking if the series is a prime series :**

1. Initialize pcheck to 1
2. Run a loop for the four terms, and initialize a variable temp with value 2
3. If the number is one or zero, pcheck = 0
4. If pcheck equals zero, loop breaks
5. Check if number is perfectly divisible by temp, and keep incrementing temp upto square root of n (Euclid’s lemma)
6. If the remainder is zero, pcheck = 0
7. If pcheck is still one after the loop ends., run another loop from the last term upto 100 next terms
8. Initialize temp to 2
9. If temp perfectly divides number, break the temp increment loop and go to next term
10. Do so until you find two such terms, which will be consecutive prime numbers after the largest term in the series
11. Return those values to the series cracker function.

**Login Module :**

Login :

1. Enter username and password
2. Check username and password from login.txt file where they are saved by opening it in read mode and scanning the file using string tokens.
3. If one of them is wrong, display message “Your username or password is wrong. Please try again.” Again take input from user.
4. When both are correct, check the user’s profile and start the corresponding menu. In this case, it is a student profile.
5. Display the menu (in case of a student) :  
   -> E-Learning  
   -> Series Cracker  
   -> Quiz  
   -> Performance Report
6. Display the menu (in case of admin profile) :  
   ->Generate performance report for a particular user  
   ->Generate performance report of users for a particular topic  
   ->Compare topics based on the performance of users in it  
   ->Add new questions for quiz  
   ->Read feedback or series suggestion messages sent by users.

Sign Up :

1. Enter username and password.
2. Check the username if it already exists. If yes, display the message “Username already exists. Please try another one.” Again take input.
3. Store the new username and password in the login.txt file by opening it in append mode.
4. Welcome the user and start the menu.