SUNNY KUMAR
16 JULY TASK

Q. Create an interactive story game where players make choices that influence the narrative. Utilize text parsing and conditional statements to build a branching storyline.

```
#include <iostream>
#include <string>

using namespace std;

// Function to display the introduction
void introduction() {
    cout << "Welcome to the Adventure Story Game!\n";
    cout << "You find yourself in a dark forest. There are two paths in front of you.\n";
    cout << "Do you choose to go left or right?\n";
}

// Function to display the first decision
void firstDecision() {
    string choice;
    cout << "Type 'left' or 'right': ";
    getline(cin, choice);

    if (choice == "left") {
        cout << "You chose to go left and you encounter a river.\n";
        cout << "Do you choose to swim across or walk along the river?\n";
        secondDecisionLeft();
    } else if (choice == "right") {
        cout << "You chose to go right and you find a cave.\n";
        cout << "Do you choose to enter the cave or continue walking?\n";
        secondDecisionRight();
    } else {
        cout << "Invalid choice. Please try again.\n";
        firstDecision();
    }
}

// Function to display the second decision if the player chose left
void secondDecisionLeft() {
    string choice;
    cout << "Type 'swim' or 'walk': ";
    getline(cin, choice);
```

```
    if (choice == "swim") {
        cout << "You swim across the river and find a treasure chest!\n";
        cout << "Congratulations, you win!\n";
    } else if (choice == "walk") {
        cout << "You walk along the river and encounter a wild animal.\n";
        cout << "Unfortunately, you couldn't escape. Game over.\n";
    } else {
        cout << "Invalid choice. Please try again.\n";
        secondDecisionLeft();
    }
}

// Function to display the second decision if the player chose right
void secondDecisionRight() {
    string choice;
    cout << "Type 'enter' or 'walk': ";
    getline(cin, choice);

    if (choice == "enter") {
        cout << "You enter the cave and find a sleeping dragon.\n";
        cout << "Do you choose to sneak past or fight the dragon?\n";
        thirdDecisionEnter();
    } else if (choice == "walk") {
        cout << "You continue walking and find a peaceful village.\n";
        cout << "You are safe now. Congratulations, you win!\n";
    } else {
        cout << "Invalid choice. Please try again.\n";
        secondDecisionRight();
    }
}

// Function to display the third decision if the player chose to enter the cave
void thirdDecisionEnter() {
    string choice;
    cout << "Type 'sneak' or 'fight': ";
    getline(cin, choice);

    if (choice == "sneak") {
        cout << "You sneak past the dragon and find a hidden exit.\n";
        cout << "Congratulations, you escape safely. You win!\n";
    } else if (choice == "fight") {
        cout << "You fight the dragon bravely but it's too strong.\n";
        cout << "Unfortunately, you couldn't defeat the dragon. Game over.\n";
```

```
    } else {
        cout << "Invalid choice. Please try again.\n";
        thirdDecisionEnter();
    }
}

int main() {
    introduction();
    firstDecision();
    return 0;
}
```

LSP (LINUX SYSTEM PROGRAMMING)

Top 20 Basic Commands with Use Cases and Exercises
While there are many commands across different operating systems and applications, here are 20 basic commands commonly found on computers:

1. dir (Windows) / ls (Linux/macOS): Lists the contents of a directory.

Use Case: You want to see all the files and folders in your current location.
Exercise: Open a terminal window (Command Prompt on Windows, Terminal on macOS/Linux) and type dir (Windows) or ls (Linux/macOS). Press Enter.
2. cd (all): Changes the current directory.

Use Case: You want to navigate to a different folder on your computer.
Exercise: Try cd Desktop (Windows/Linux/macOS) to navigate to your Desktop folder. Then use dir (Windows) or ls (Linux/macOS) to see the contents.
3. mkdir (all): Creates a new directory.

Use Case: You want to organize your files by creating a new folder.
Exercise: Use mkdir Documents (Windows/Linux/macOS) to create a new folder named "Documents". Then use dir (Windows) or ls (Linux/macOS) to see if it's there.
4. rm (Linux/macOS) / del (Windows): Deletes a file or directory (use with caution!).

Use Case: You want to remove an unwanted file or folder.
Exercise: Important: Never delete anything critical! In a safe space (like a temporary folder), create a text file named "test.txt" and then use rm test.txt (Linux/macOS) or del test.txt (Windows) to delete it.
5. copy (Windows) / cp (Linux/macOS): Copies a file.

Use Case: You want to duplicate a file to another location.

Exercise: Create another text file named "test2.txt". Use copy test.txt test2.txt (Windows) or cp test.txt test2.txt (Linux/macOS) to copy "test.txt" as "test2.txt".
6. move (Windows) / mv (Linux/macOS): Moves a file from one location to another.

Use Case: You want to organize your files by moving them to a different folder.
Exercise: Use move test2.txt Documents (Windows) or mv test2.txt Documents (Linux/macOS) to move "test2.txt" to the "Documents" folder (assuming it exists).
7. rename (Windows) / mv (Linux/macOS): Renames a file.

Use Case: You want to give a file a different name.
Exercise: Use rename test.txt newname.txt (Windows) or mv test.txt newname.txt (Linux/macOS) to rename "test.txt" to "newname.txt".
8. ping (all): Checks if another computer is reachable on a network.

Use Case: You want to see if you can connect to a website or another device.
Exercise: Use ping google.com (all) to see if you can reach Google's servers.
9. ipconfig (Windows) / ifconfig (Linux/macOS): Shows network configuration information.

Use Case: You want to troubleshoot network connectivity issues.
Exercise: Use ipconfig (Windows) or ifconfig (Linux/macOS) to see your IP address and other network details.
10. help (all): Provides help information for other commands.

Use Case: You're unsure about how to use a specific command.
Exercise: If you're stuck on command like mv, type help mv (all) to see a manual page with usage information.
11. clear (all): Clears the screen (text) in the terminal window.

Use Case: Your terminal window is cluttered with previous commands, and you want a clean slate.
Exercise: Type clear (all) to clear the screen.
12. date (all): Shows the current date and time.
14. time (all): (continued) You want to see how long a command takes to execute.

Exercise: Try time ls (all) to see how long it takes to list the directory contents.
15. mkdir -p (Linux/macOS): Creates a directory and any missing parent directories.

Use Case: You want to create a new folder within a nested structure that might not exist yet.
Exercise: Use mkdir -p Documents/Subfolder1/Subfolder2 (Linux/macOS) to create "Subfolder2" within "Subfolder1" inside the "Documents" folder (assuming "Documents" exists).
16. cat (Linux/macOS): Displays the contents of a text file.

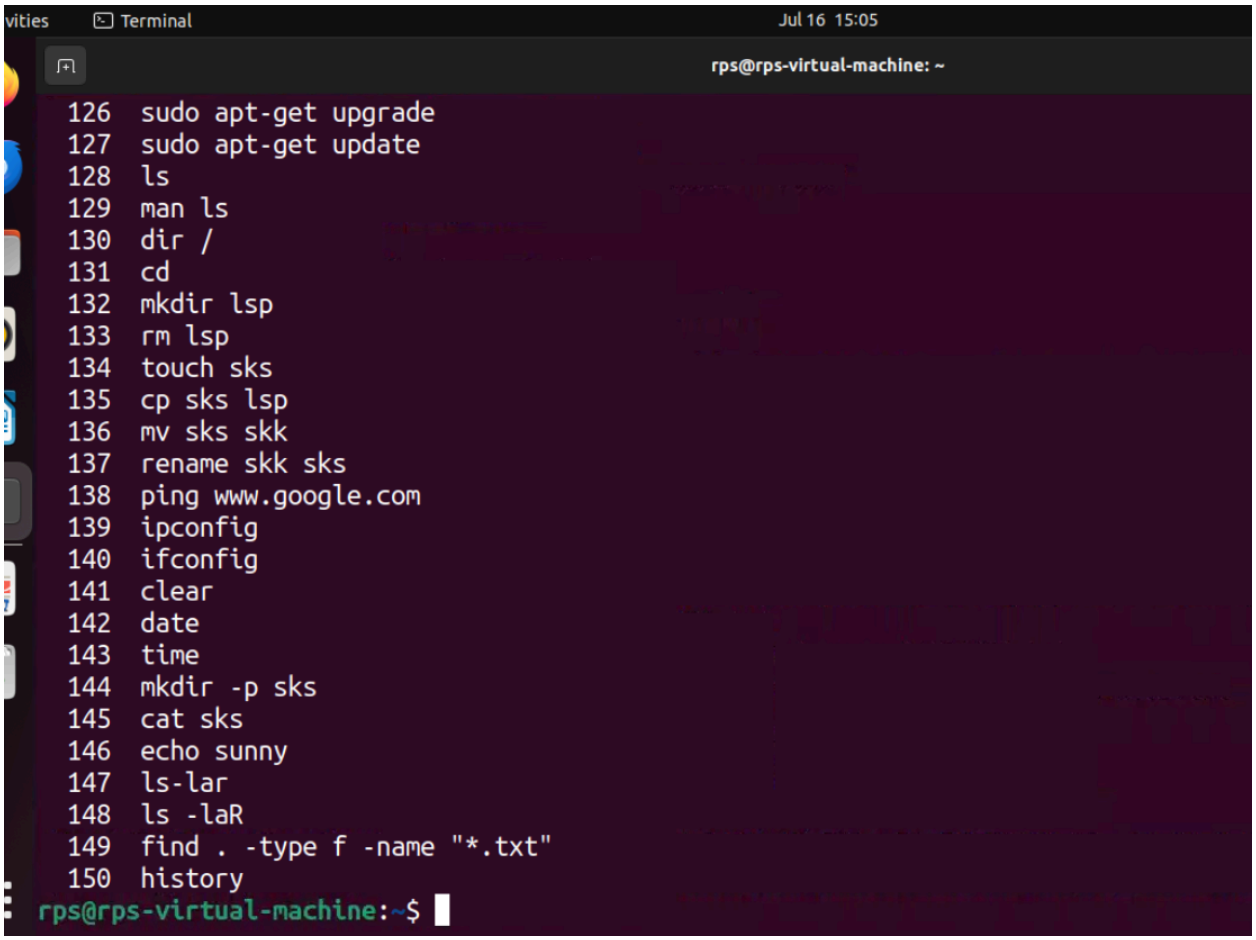
Use Case: You want to read the contents of a text file without opening it in a separate program.

Exercise: Create a text file with some content and use cat filename.txt (Linux/macOS) to see its contents.
17. echo (all): Prints text to the terminal window.

Use Case: You want to display a message or variable in the terminal.
Exercise: Use echo Hello, world! (all) to print the message to the screen.
18. sudo (Linux/macOS): Grants temporary superuser privileges to execute a command (use with caution!).

Use Case: You need to perform an action that requires administrative rights.
Exercise: Important: Never use sudo for untrusted commands! In a safe scenario (like creating a test file), use sudo touch important.txt to create a file that might require admin access (assuming you have the password).
19. shutdown (Linux/macOS) / shutdown /s /t (Windows): Initiates a system shutdown or restart.

Use Case: You want to turn off or restart your computer.
Exercise: Important: Don't accidentally shut down your computer! This is for learning purposes only. Look up the specific options for your system to safely test a shutdown with a delay (e.g., shutdown /s /t 60 for Windows to shutdown in 60 seconds).
20. history (all): Shows a list of previously entered commands.



Q. Write a command using ls to list all files (including hidden files) in the current directory and its subdirectories.

Ls-laR

Q.Modify the previous command to display only files with a specific extension (e.g., .txt).
find . -type f -name "*.txt"