Lab 1: Write on Multimedia Basic Software Tools

Multimedia Basic Software Tools
Multimedia software tools are essential for designing, modifying, and showcasing multimedia content such as text, images, audio, video, and animation. These tools are grouped based on their functionalities. Below are the primary types of software commonly utilized in multimedia applications:-

1. Text Editing Tools
Text is a vital part of multimedia content. Text editors assist in creating and formatting written material, offering options like font adjustment, alignment, spell checking, grammar support, and hyperlink insertion. Many modern tools also support shared editing, cloud storage, and document collaboration.

Examples: Microsoft Word, Notepad, Google Docs
Key Features: Font adjustment, text styling, spell check, hyperlinking, shared editing, and cloud-based access.

2. Image Editing Tools
Image editing software allows users to design, refine, and improve digital pictures. Key functionalities include cropping, resizing, layering, adjusting colors, removing backgrounds, and applying various visual effects. Some tools now feature AI capabilities for automated editing and object removal.

Examples: Adobe Photoshop, GIMP, CorelDraw
Key Features: Cropping, resizing, layering, color enhancement, visual effects, AI-driven tools, and vector support.

3. Audio Editing Tools
These tools are used to capture and modify audio files. They include features like multiple track editing, noise suppression, pitch adjustment, voice enhancement, and applying effects in real time. They are frequently used in audio production, podcasting, and sound editing.

Examples: Audacity, Adobe Audition, GarageBand
Key Features: Noise suppression, audio mixing, voice refinement, multi-track support, real-time effects, and sound equalization.

4. Video Editing Tools
Video editing software enables users to trim, arrange, and enhance video files. Features may include transitions, effects, color adjustments, motion tracking, and audio synchronization. Many advanced tools also support ultra-high-definition formats like 4K and 8K.

Examples: Adobe Premiere Pro, Final Cut Pro, DaVinci Resolve
Key Features: Trimming, visual effects, transitions, color correction, motion tracking, audio syncing, and 4K/8K support.

5. Presentation Tools

Presentation tools are used to build and deliver multimedia-enriched slides. These tools provide features like slide transitions, integrated media, pre-designed templates, and animations. Many also support online collaboration and interactive elements for better engagement.

Examples: Microsoft PowerPoint, Google Slides, Prezi
Key Features: Slide transitions, media integration, templates, animations, cloud collaboration, and interactive features.

Lab 2: What are the types of data compression technique?

1. Lossless Compression
Lossless compression minimizes file size without discarding any data. When decompressed, the original data is fully recovered without any alterations. This method is crucial for scenarios where complete data accuracy is vital, such as in software files, documents, or high-fidelity image storage.

Examples:-
Huffman Coding: Encodes data by analyzing frequency patterns.
Lempel-Ziv-Welch (LZW): Common in image formats like GIF and TIFF.
Deflate: Applied in ZIP archives, PNG images, and Gzip files.
Run-Length Encoding (RLE): Compresses sequences of repeating values (used in TIFF files).

Key Features:-
Original data is entirely recoverable after decompression.
Commonly used for documents, programs, and lossless image formats like PNG.

2. Lossy Compression
Lossy compression achieves smaller file sizes by removing non-essential data, which results in some loss of quality. It's suitable for multimedia content where slight quality degradation is acceptable and often unnoticeable. The primary aim is to balance file size reduction with acceptable visual or auditory quality.

Examples:-
JPEG: A commonly used lossy image format for web graphics.
MP3: A popular format for compressing music and spoken audio.
MPEG (H.264, H.265): Widely used video compression standards for online streaming and media storage.

Key Features:-
Substantial file size reduction.
Minor quality loss that is typically unnoticeable to users.
Preferred for compressing videos, images, and audio files.

Lab 3: Compare the file sizes of an uncompressed BMP image and a compressed JPEG image of the same resolution. Explain why the sizes differ and the impact on quality.

BMP (Bitmap Image File Format)
The BMP format, developed by Microsoft, stores image data in an uncompressed form. It retains every pixel's color value exactly as it is, making it highly accurate in preserving image details and quality. Because it doesn't use any compression technique, BMP files tend to be significantly larger in size. They are often used in professional environments for editing, printing, and archival where high fidelity is required.

JPEG (Joint Photographic Experts Group)
JPEG is a widely-used compressed image format that employs lossy compression to minimize file size. It utilizes Discrete Cosine Transform (DCT) to eliminate visually insignificant details. This results in smaller files, but at the cost of some image quality, depending on the compression level. JPEG is ideal for photography, web graphics, and general-purpose image sharing due to its space efficiency.

Why Do BMP and JPEG Have Different File Sizes?

BMP Files:
BMP images store raw, uncompressed pixel data. In a 24-bit image, each pixel uses 3 bytes (one for red, green, and blue). As resolution and color depth increase, so does the file size.

Result: BMP files are considerably larger, especially for high-resolution images.

JPEG Files:
JPEGs achieve smaller sizes by permanently removing data that is less likely to be noticed by the viewer. This includes subtle color transitions and fine textures.

Result: JPEG images are much more compact in size, though they may lose some quality depending on compression settings.

Impact on Image Quality

BMP: Since the BMP format saves every pixel's exact value, there is zero loss in quality. It is ideal when image precision is critical, such as in editing and printing.

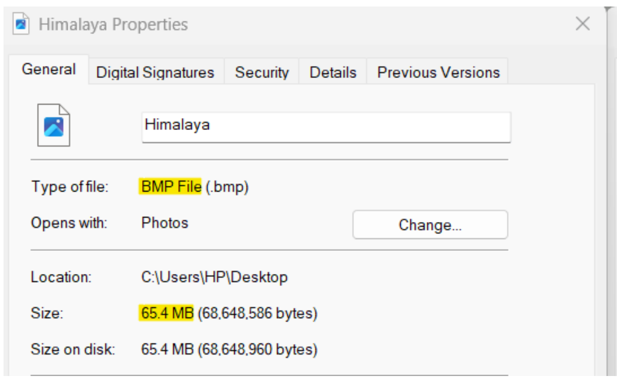
JPEG: The lossy nature of JPEG compression leads to a drop in image quality. High compression levels may introduce visible defects like blurring and pixilation. However, with low compression (high quality), the image still appears clear while saving space.

Comparison of File Sizes and Quality.

Feature	BMP (Uncompressed)	JPEG (Compressed)
File Size	Very large (~6 MB for 1920×1080)	Much smaller (~50 KB to 2 MB for same size)
Compression Type	None (Uncompressed)	Lossy Compression
Image Quality	No loss (Perfect quality)	Lossy (Depends on compression level)
Best For	Image editing, archiving, professional use	Web, photography, and sharing
Compression Artifacts	None	Yes (e.g., blockiness, blurring, noise)
Storage Efficiency	Poor (large file size)	Excellent (small file size)
Editing Flexibility	Excellent (since no loss occurs)	Poor (quality degrades with each edit)

Example Comparison:

Here, two video files in different formats are compared. The highlighted file sizes demonstrate how different video codecs and resolutions influence the final file size and storage efficiency.

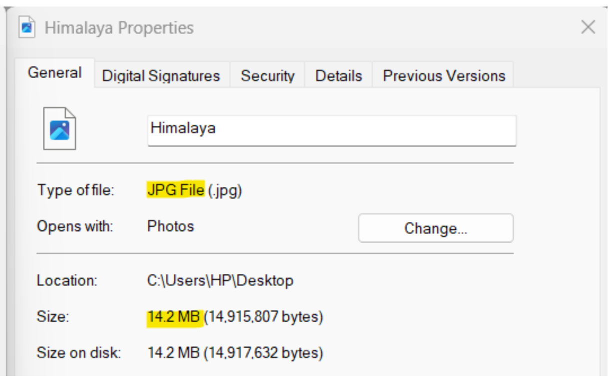


Himalaya Properties

General | Digital Signatures | Security | Details | Previous Versions

Himalaya

Type of file: BMP File (.bmp)
Opens with: Photos
Location: C:\Users\HP\Desktop
Size: 65.4 MB (68,648,586 bytes)
Size on disk: 65.4 MB (68,648,960 bytes)



Himalaya Properties

General | Digital Signatures | Security | Details | Previous Versions

Himalaya

Type of file: JPG File (.jpg)
Opens with: Photos
Location: C:\Users\HP\Desktop
Size: 14.2 MB (14,915,807 bytes)
Size on disk: 14.2 MB (14,917,632 bytes)

Lab 4: Record an audio clip in WAV format (uncompressed) and in MP3 format (compressed). Analyze the file size and discuss the trade-off between file size and audio quality.

WAV Format (Uncompressed)

File Size: Approximately 229 MB for one hour of CD-quality audio (44.1 kHz, 16-bit, stereo).
Audio Quality: WAV files store audio without compression, ensuring every detail of the original recording is preserved. This makes them perfect for use cases where sound fidelity is critical, such as music production, editing, or archival storage. Since there is no data loss, WAV files are free from compression artifacts and retain exact audio accuracy.
Trade Off: The main limitation is the large file size, which can be problematic when dealing with limited storage space or when sharing over slower networks.

Best Use: Ideal for professional audio editing, mastering, or long-term preservation where sound quality is a top priority.

MP3 Format (Compressed)

File Size: Around 20.7 MB for the same 1-hour audio file encoded at 128 kbps.
Audio Quality: MP3 applies lossy compression, meaning some parts of the audio are removed to shrink file size. The resulting sound quality depends on the bitrate used:

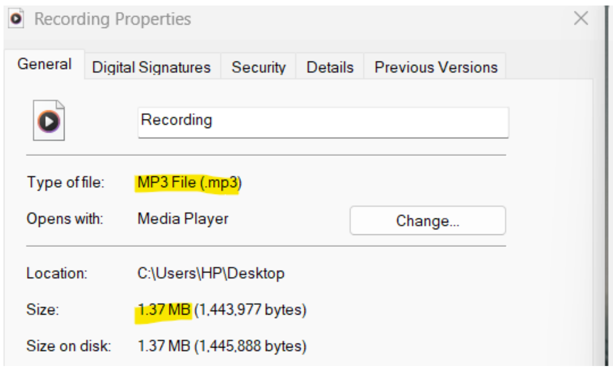
> **128 kbps:** Some quality degradation is noticeable, especially in high-frequency sounds or complex tracks.
> **256 kbps / 320 kbps:** Offers better sound with reduced artifacts, though still not identical to uncompressed formats.

Advantage: MP3 greatly reduces file size, making it highly efficient for saving space, sharing files, and streaming audio online. It's suitable for everyday use where perfect fidelity isn't necessary.
Disadvantage: Some audio information is permanently lost due to compression. This can become evident on high-end audio equipment or in critical listening environments.

Best Use: General audio playback, distribution, and web streaming where storage savings are more important than absolute audio precision.

Example Comparison:

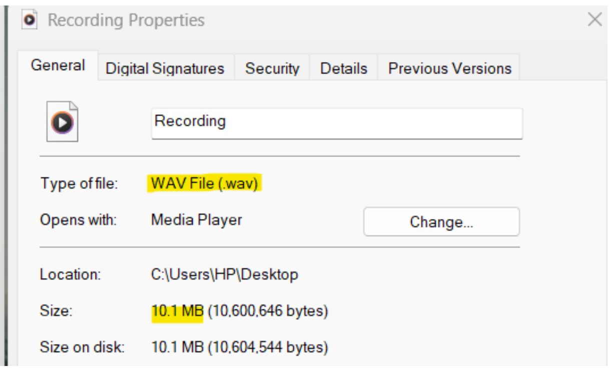
Below are the properties of two audio files in different formats. The highlighted file sizes show how compression and encoding techniques vary across formats, affecting the overall storage space required.



Recording Properties

Recording

Type of file: MP3 File (.mp3)
Opens with: Media Player
Location: C:\Users\HP\Desktop
Size: 1.37 MB (1,443,977 bytes)
Size on disk: 1.37 MB (1,445,888 bytes)



Recording Properties

Recording

Type of file: WAV File (.wav)
Opens with: Media Player
Location: C:\Users\HP\Desktop
Size: 10.1 MB (10,600,646 bytes)
Size on disk: 10.1 MB (10,604,544 bytes)

Lab 5: Create a multimedia presentation (using PowerPoint or Prezi) on "The Impact of Multimedia in Education." Include text, images, audio, video, and animations.

Below is a step-by-step guide for developing a multimedia-rich presentation in PowerPoint titled *"The Impact of Multimedia in Education"*:

Step 1: Launch PowerPoint and Select a Design
Open Microsoft PowerPoint.
Choose **Blank Presentation** or browse through **Design > Themes** to apply a polished layout to your slides.

Step 2: Design the Title Slide
Use the first slide to display the main title: *"The Impact of Multimedia in Education"*. Add a subtitle such as *"Enhancing Learning through Technology"*. Include a relevant image illustrating multimedia use in the classroom or digital education.

Step 3: Insert Text Content and Visuals
Create a new slide using **Home > New Slide**.
Choose the "Title and Content" layout to present major points.
Add supporting visuals by selecting **Insert > Pictures** and choosing appropriate educational graphics.

Step 4: Add Audio Narration
Navigate to **Insert > Audio**, then select **Record Audio** or **Audio on My PC**.
Record a brief explanation or commentary about the slide content.
Position the speaker icon neatly, preferably at the bottom or in a corner.

Step 5: Embed Video Clips
Go to **Insert > Video**, then pick **Online Video** or **Video on My PC**.
Insert a short educational video related to multimedia's influence on teaching and learning.
Adjust the video's size and placement to keep your slide visually balanced.

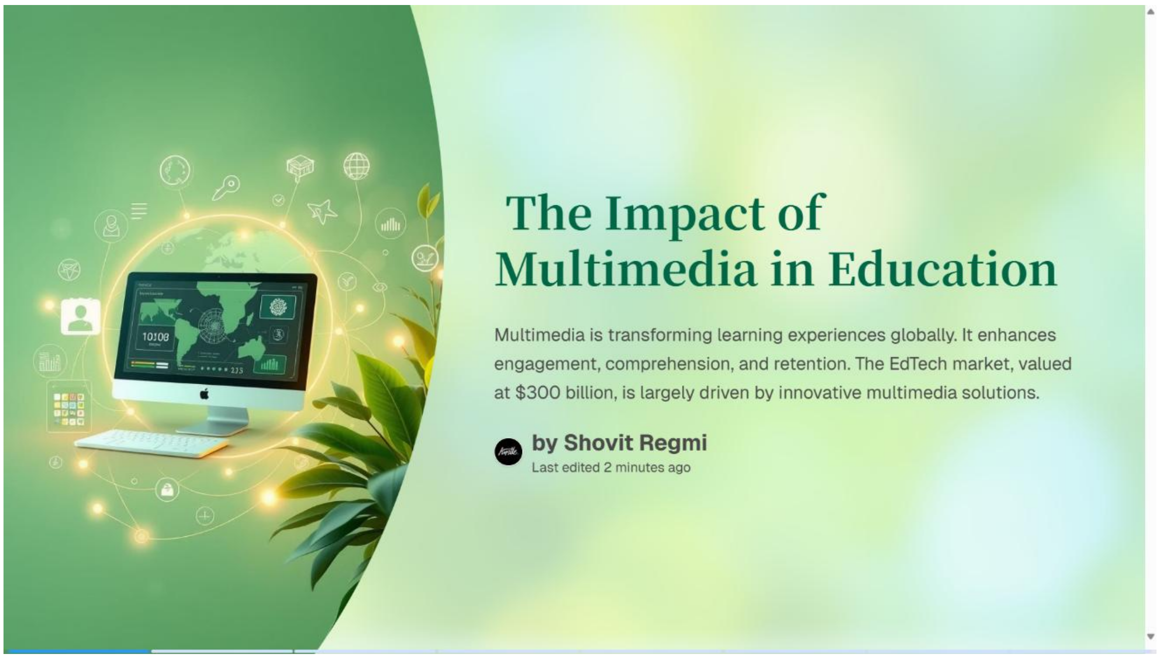
Step 6: Apply Animations and Slide Transitions
Select any text or image box and click on the **Animations** tab.
Use effects like **Fade**, **Appear**, or **Fly In** for engaging motion.
Then, under **Transitions**, apply transitions such as **Morph**, **Push**, or **Wipe** for smoother slide changes.

Step 7: Incorporate Interactive Features
Use **Insert > Link** to add hyperlinks for quick navigation between slides.
You can also insert buttons or shapes linked to specific slides to make the presentation interactive.

Step 8: Final Review and File Export
Play the entire presentation by selecting **Slide Show > From Beginning** to ensure animations and media
Work as intended.

Save your project as a **.pptx** file, and if needed, export it as a video file (**.mp4**) via **File > Export > Create a Video**.

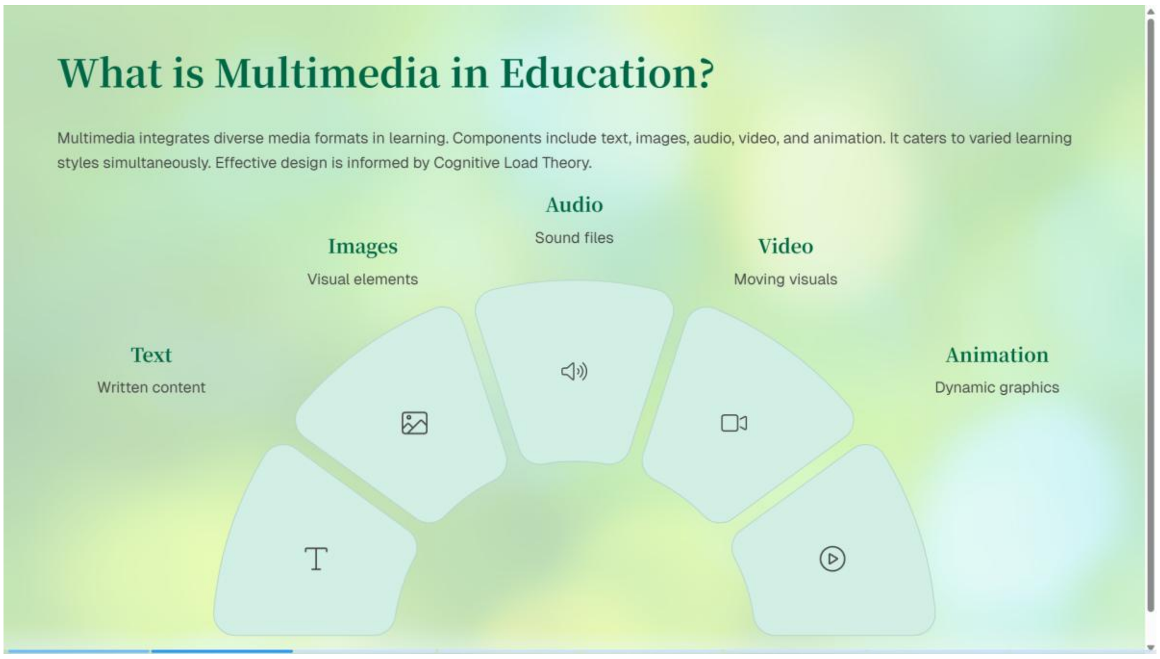
Example:



The Impact of Multimedia in Education

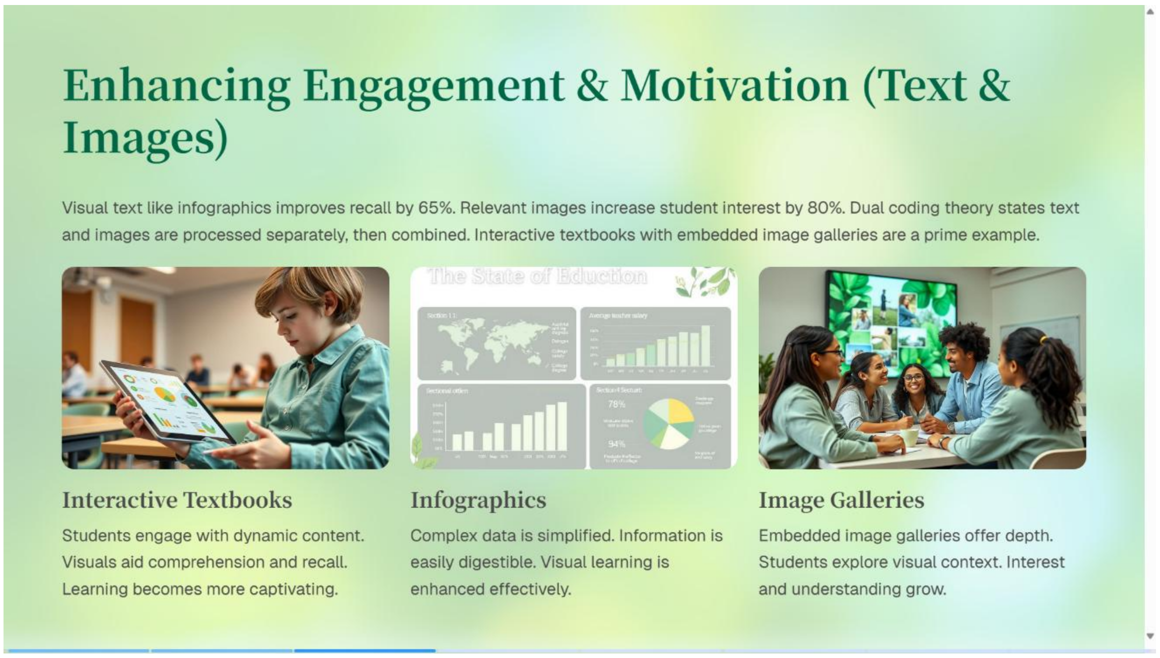
Multimedia is transforming learning experiences globally. It enhances engagement, comprehension, and retention. The EdTech market, valued at $300 billion, is largely driven by innovative multimedia solutions.

by Shovit Regmi
Last edited 2 minutes ago



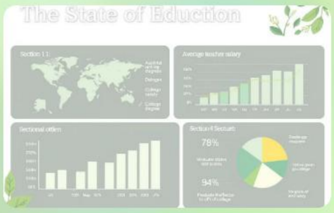
What is Multimedia in Education?

Multimedia integrates diverse media formats in learning. Components include text, images, audio, video, and animation. It caters to varied learning styles simultaneously. Effective design is informed by Cognitive Load Theory.



Enhancing Engagement & Motivation (Text & Images)

Visual text like infographics improves recall by 65%. Relevant images increase student interest by 80%. Dual coding theory states text and images are processed separately, then combined. Interactive textbooks with embedded image galleries are a prime example.



Interactive Textbooks

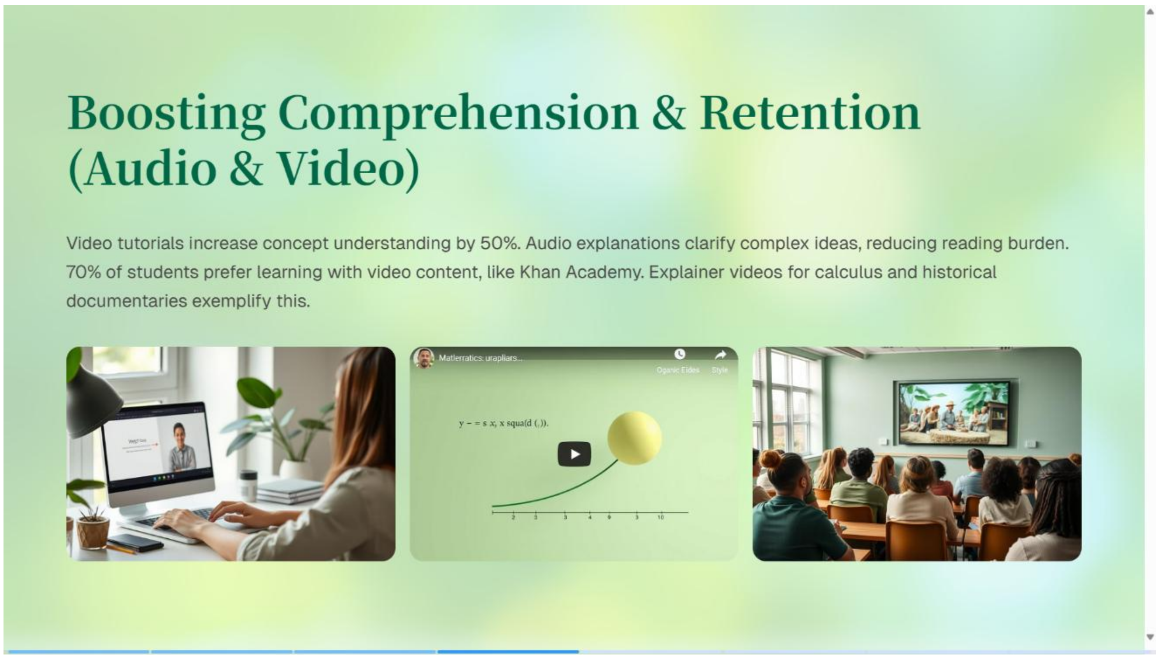
Students engage with dynamic content. Visuals aid comprehension and recall. Learning becomes more captivating.

Infographics

Complex data is simplified. Information is easily digestible. Visual learning is enhanced effectively.

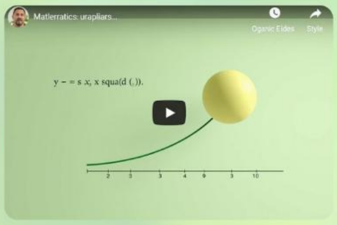
Image Galleries

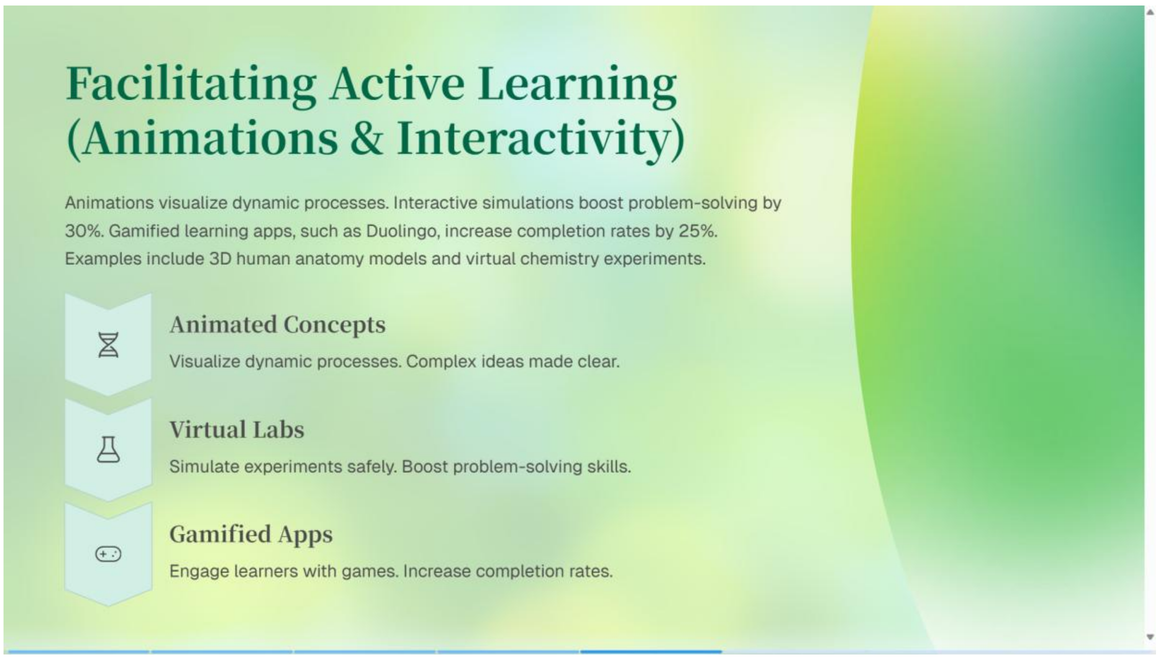
Embedded image galleries offer depth. Students explore visual context. Interest and understanding grow.



Boosting Comprehension & Retention (Audio & Video)

Video tutorials increase concept understanding by 50%. Audio explanations clarify complex ideas, reducing reading burden. 70% of students prefer learning with video content, like Khan Academy. Explainer videos for calculus and historical documentaries exemplify this.





Facilitating Active Learning (Animations & Interactivity)

Animations visualize dynamic processes. Interactive simulations boost problem-solving by 30%. Gamified learning apps, such as Duolingo, increase completion rates by 25%. Examples include 3D human anatomy models and virtual chemistry experiments.

Animated Concepts

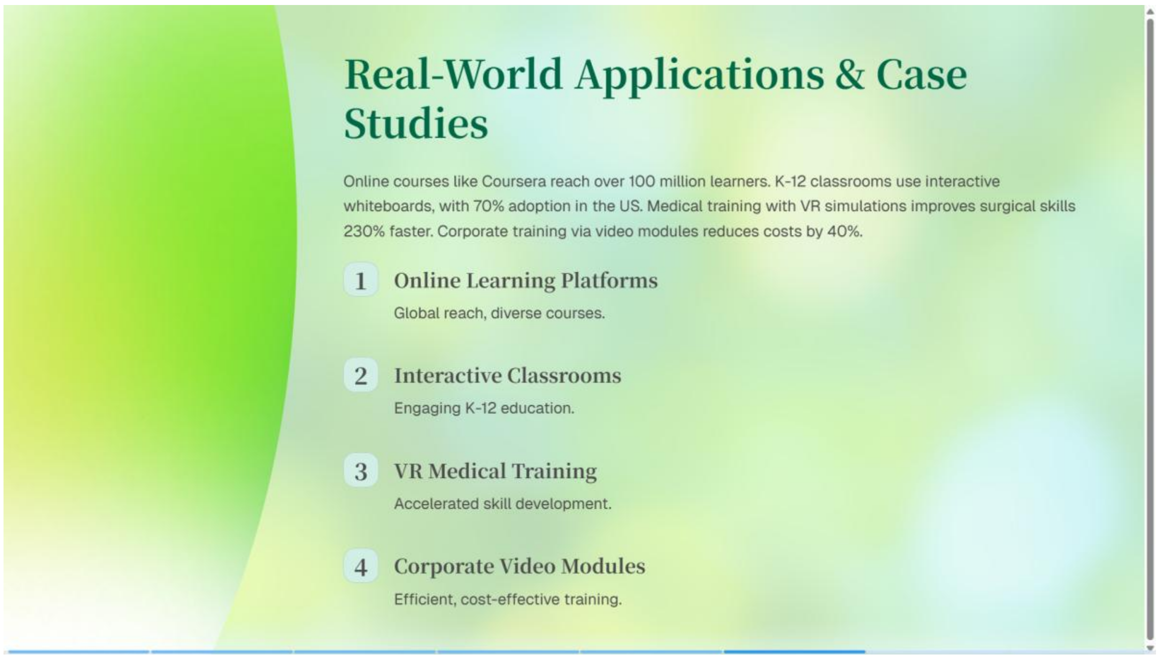
Visualize dynamic processes. Complex ideas made clear.

Virtual Labs

Simulate experiments safely. Boost problem-solving skills.

Gamified Apps

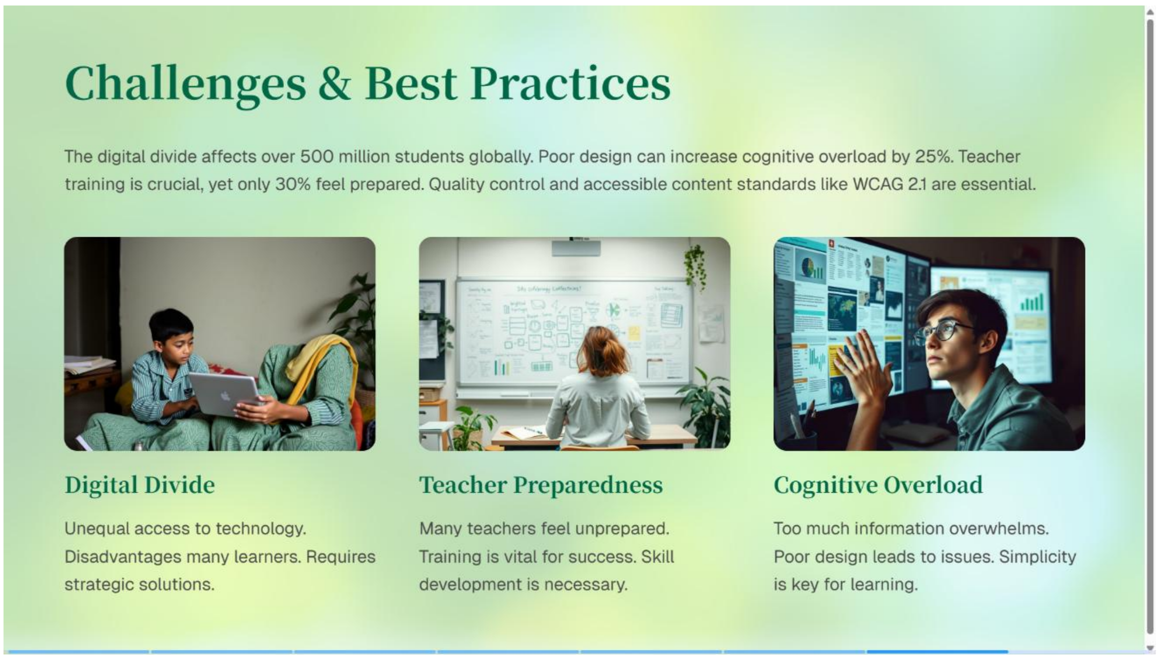
Engage learners with games. Increase completion rates.



Real-World Applications & Case Studies

Online courses like Coursera reach over 100 million learners. K-12 classrooms use interactive whiteboards, with 70% adoption in the US. Medical training with VR simulations improves surgical skills 230% faster. Corporate training via video modules reduces costs by 40%.

1. **Online Learning Platforms**
   Global reach, diverse courses.
2. **Interactive Classrooms**
   Engaging K-12 education.
3. **VR Medical Training**
   Accelerated skill development.
4. **Corporate Video Modules**
   Efficient, cost-effective training.



Challenges & Best Practices

The digital divide affects over 500 million students globally. Poor design can increase cognitive overload by 25%. Teacher training is crucial, yet only 30% feel prepared. Quality control and accessible content standards like WCAG 2.1 are essential.

Digital Divide

Unequal access to technology. Disadvantages many learners. Requires strategic solutions.

Teacher Preparedness

Many teachers feel unprepared. Training is vital for success. Skill development is necessary.

Cognitive Overload

Too much information overwhelms. Poor design leads to issues. Simplicity is key for learning.

Conclusion: The Future of Learning

Multimedia is indispensable for modern education. It drives personalized, engaging, and accessible learning. Continued innovation in AI and VR will redefine possibilities. Investment in quality multimedia yields significant educational returns.

Indispensable Tool

Essential for modern learning.

Personalized Learning

Tailored educational experiences.

Future Innovation

AI and VR redefine possibilities.

Significant Returns

Investment yields benefits.

Lab 6: Record a short video and embed it into a PowerPoint presentation. Add captions and transitions to enhance the presentation.

Here are the step by step method to capture a video and add it into a presentation to enhance user interaction and engagement.

Step 1: Capture a Short Video

Method 1: Using PowerPoint's Built-in Video Recorder
Open your PowerPoint presentation and navigate to the desired slide.
Go to **Insert > Video > Record Video** (available in newer versions).
A recording interface will open—click **Record**, deliver your message, then click **Stop**.
Click **Insert** to embed the recorded video into the slide.

Method 2: Using an External Device
Use a smartphone or webcam to film your video.
Save the file in **MP4 format** for best compatibility.
Transfer the video to your computer for use in the presentation.

Step 2: Insert the Video into PowerPoint

Open your PowerPoint file and go to the slide where the video should appear.
Click **Insert > Video > Video on My PC**.
Select your video file and click **Insert**.
Resize and move the video as needed to fit the slide layout.

Step 3: Add Captions for Accessibility

Click the embedded video to select it.
Go to **Playback > Insert Captions**.
Choose **Add Captions**, then click **Create New Captions**.
While playing the video, type the corresponding text to match the audio.
Save the caption file when complete.

Step 4: Apply Transitions for a Smooth Flow

Select the video slide, then go to the **Transitions** tab.
Choose a visual effect like **Fade**, **Morph**, or **Push**.
Adjust the **Duration** to control how smoothly the slide appears.
Click **Apply to all** if you'd like to use the same transition throughout the presentation.

Step 5: Preview and Save Your Work

Use **Slide Show > From Current Slide** to test the video playback and transitions.
Save your project as a **.pptx** file.
If you want to convert the presentation into a video, go to **File > Export > Create a Video**, and save it in **MP4 format**.

Example:

Lab 7: Using Adobe Photoshop, design a poster for a product or event.

Include:

Layer adjustments (background, text, and images), Image retouching tools (Clone Stamp, Healing Brush), Text effects (shadow, stroke, and gradient), Final export as PNG and JPEG—compare the file sizes and quality.

Introduction:

Adobe Photoshop is a powerful image editing software widely used by designers, photographers, and digital artists across the globe. It provides a vast array of tools for editing images, retouching, and creating visual designs. Due to its flexibility and precision, Photoshop is a go-to application for both beginners and professionals when it comes to poster creation, image enhancement, and digital art.

Steps:

1. Setting up Your Document

Launch Adobe Photoshop and navigate to File > New.
Select a common poster dimension, such as 1920 x 1080 pixels or A4 size (210 x 297 mm).
Set the resolution to 300 DPI for print or 72 DPI for digital display.
Choose RGB color mode for digital designs and CMYK for print materials.

2. Creating the Background

Click on the Background Layer and apply a gradient fill, solid color, or insert a background image.
Enhance the background using adjustment layers like Brightness/Contrast or Hue/Saturation.

3. Inserting and Editing Images

Add product or event images by going to File > Place Embedded.
Use tools like the Clone Stamp or Healing Brush to correct flaws and eliminate unwanted areas.
Modify the image’s opacity or blend mode to integrate it smoothly with the background.

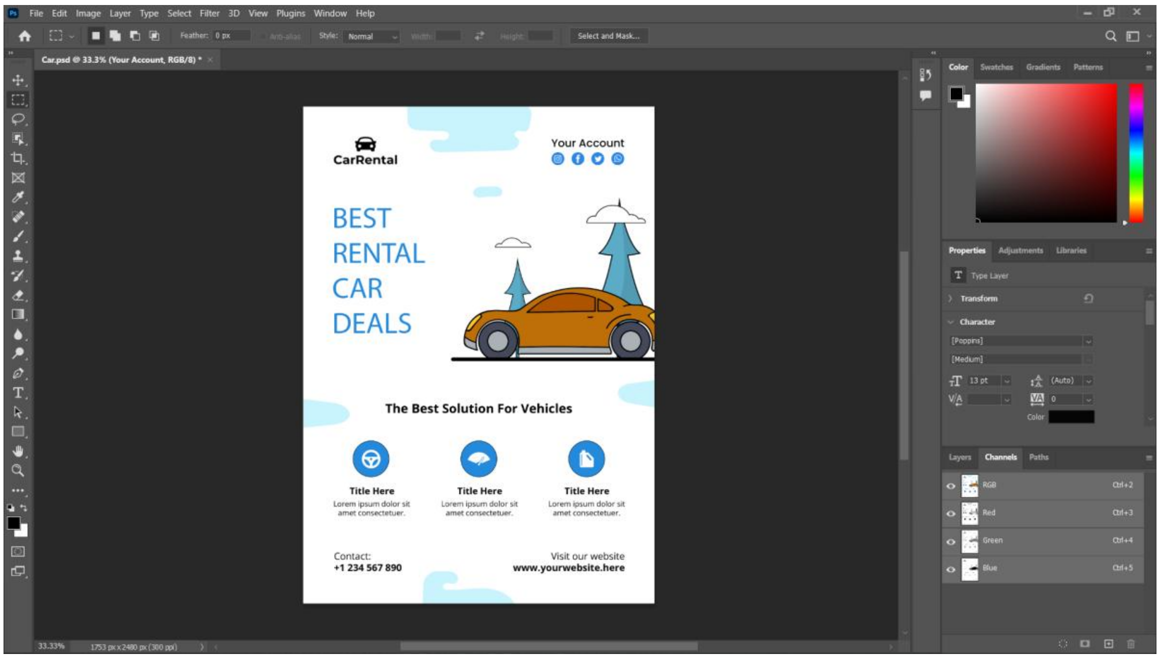
4. Adding Text and Applying Effects

Use the Text Tool (T) to insert the event name or product information.
Enhance the text using these effects:
Drop Shadow: Add depth via Layer Style > Drop Shadow.
Stroke: Outline the text using Layer Style > Stroke.
Gradient Overlay: Apply stylish gradients with Layer Style > Gradient Overlay.

5. Final Adjustments and Export

Refine the layer layout, adjust colors, and make final tweaks for a clean, balanced design.
Export your poster by going to File > Export > Save for Web (Legacy).
Choose **JPEG** for a compressed, smaller-sized file (with some quality loss).
Choose **PNG** for higher quality and transparency support (lossless compression).
Compare the file sizes and visual quality of both formats to understand the trade-offs.

Example:


File Edit Image Layer Type Select Filter 3D View Plugins Window Help
Select and Mask...
Car.psd @ 33.3% (Your Account, RGB/8) *
CarRental
Your Account
BEST
RENTAL
CAR
DEALS
The Best Solution For Vehicles
Title Here
Title Here
Title Here
Color Swatches Gradients Patterns
Properties Adjustments Libraries
Type Layer
Transform
Character
Layers Channels Paths
RGB Ctrl+2
Red Ctrl+3
Green Ctrl+4
Blue Ctrl+5

Feather: 0 px

Style: Normal

Lorem ipsum dolor sit amet consectetuer.

Lorem ipsum dolor sit amet consectetuer.

Lorem ipsum dolor sit amet consectetuer.

Contact:
+1 234 567 890

Visit our website
www.yourwebsite.here

[Poppins]

[Medium]

13 pt (Auto)

0

33.33% 1753 px x 2480 px (300 ppi)

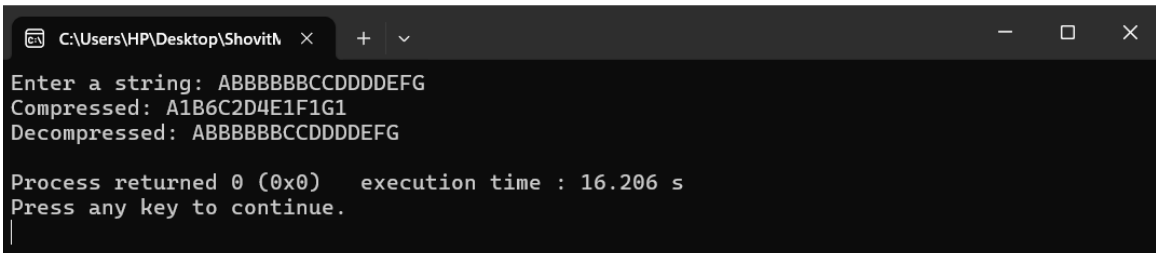
Lab 8: WAP for File Compression using Run Length Encoding.

Code:

```
#include <stdio.h>
#include <string.h>
#define MAX 1000

void rle_compress(char *input, char *output)
{
   int count, j = 0;
   int i;      for (i = 0; input[i] != '\0'; i++)
 {
       count = 1;
       while (input[i] == input[i + 1])
           {
           count++;
           i++;
            }
       output[j++] = input[i];
       j += sprintf(output + j, "%d", count);
   }
   output[j] = '\0';
}
void rle_decompress (char *input, char *output)
{
   int j = 0, count;
   int i, k;  // Moved declarations here
   for (i = 0; input[i] != '\0'; i++)
           {
       char ch = input[i];
       count = 0;
       while (input[i + 1] >= '0' && input[i + 1] <= '9')
                       {
           count = count * 10 + (input[i + 1] - '0');
           i++;
                        }
       for (k = 0; k < count; k++)
                       {
           output[j++] = ch;
                       }
           }
   output[j] = '\0';
 }

int main()
{
   char input[MAX], compressed[MAX], decompressed[MAX];
   printf("Enter a string: ");
   scanf("%s", input);
   rle_compress(input, compressed);
```

```
    printf("Compressed: %s\n", compressed);
    rle_decompress(compressed, decompressed);
    printf("Decompressed: %s\n", decompressed);
    return 0;
}
```

Output:



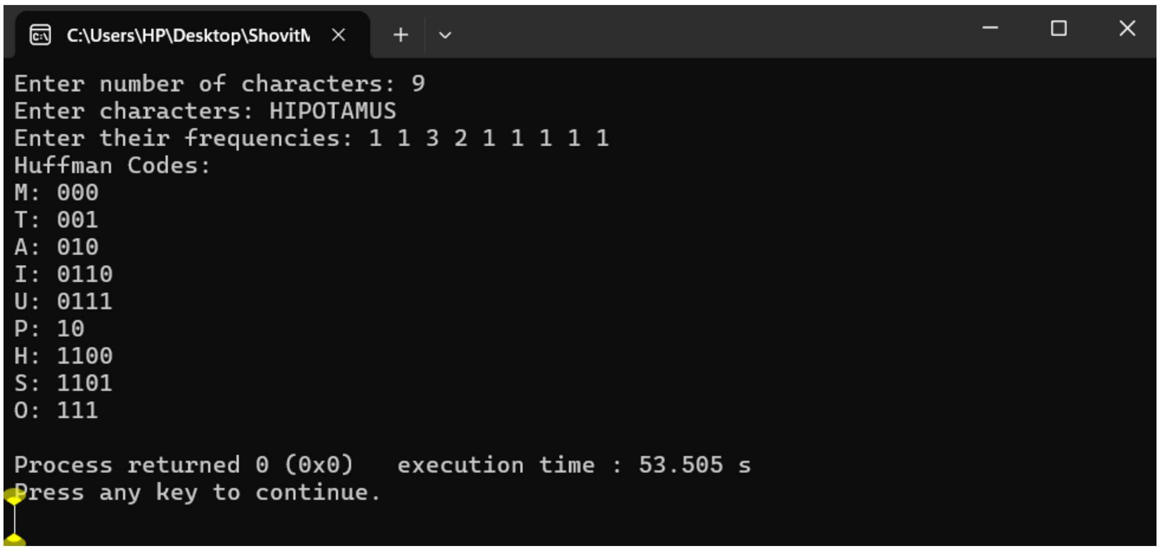
```
Enter a string: ABBBBBBCCDDDDEFG
Compressed: A1B6C2D4E1F1G1
Decompressed: ABBBBBBCCDDDDEFG

Process returned 0 (0x0)   execution time : 16.206 s
Press any key to continue.
```

Lab 9: WAP to build a Huffman Tree and print the codes.

Code:

```
#include <stdio.h>
#include <stdlib.h>

struct HuffmanNode {
    char data;
    int freq;
    struct HuffmanNode *left, *right;
};

struct MinHeap {
    int size;
    struct HuffmanNode **array;
};

struct HuffmanNode* createNode(char data, int freq) {
    struct HuffmanNode* node = (struct HuffmanNode*)malloc(sizeof(struct HuffmanNode));
    node->data = data;
    node->freq = freq;
    node->left = node->right = NULL;
    return node;
}

void swapNodes(struct HuffmanNode** a, struct HuffmanNode** b) {
    struct HuffmanNode* temp = *a;
    *a = *b;
    *b = temp;
}

void minHeapify(struct MinHeap* minHeap, int index) {
    int smallest = index;
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    if (left < minHeap->size && minHeap->array[left]->freq < minHeap->array[smallest]-
>freq)
        smallest = left;
    if (right < minHeap->size && minHeap->array[right]->freq < minHeap->array[smallest]-
>freq)
        smallest = right;
    if (smallest != index) {
        swapNodes(&minHeap->array[smallest], &minHeap->array[index]);
        minHeapify(minHeap, smallest);
    }
}

struct HuffmanNode* extractMin(struct MinHeap* minHeap) {
    struct HuffmanNode* temp = minHeap->array[0];
    minHeap->array[0] = minHeap->array[minHeap->size - 1];
    minHeap->size--;
    minHeapify(minHeap, 0);
    return temp;
}

void insertMinHeap(struct MinHeap* minHeap, struct HuffmanNode* node) {
    minHeap->size++;
    int i = minHeap->size - 1;
    while (i && node->freq < minHeap->array[(i - 1) / 2]->freq) {
        minHeap->array[i] = minHeap->array[(i - 1) / 2];
        i = (i - 1) / 2;
    }
    minHeap->array[i] = node;
}
struct MinHeap* createAndBuildMinHeap(char data[], int freq[], int size) {
    struct MinHeap* minHeap = (struct MinHeap*)malloc(sizeof(struct MinHeap));
    minHeap->size = size;
    minHeap->array = (struct HuffmanNode**)malloc(size * sizeof(struct HuffmanNode*));
    for (int i = 0; i < size; i++)
        minHeap->array[i] = createNode(data[i], freq[i]);
    for (int i = (size - 1) / 2; i >= 0; i--)
        minHeapify(minHeap, i);

    return minHeap;
}

struct HuffmanNode* buildHuffmanTree(char data[], int freq[], int size) {
    struct MinHeap* minHeap = createAndBuildMinHeap(data, freq, size);
    while (minHeap->size > 1) {
        struct HuffmanNode* left = extractMin(minHeap);
        struct HuffmanNode* right = extractMin(minHeap);
        struct HuffmanNode* top = createNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;
        insertMinHeap(minHeap, top);
    }
    return extractMin(minHeap);
}

void printCodes(struct HuffmanNode* root, int arr[], int top) {
    if (root->left) {
        arr[top] = 0;
        printCodes(root->left, arr, top + 1);
    }
    if (root->right) {
        arr[top] = 1;
        printCodes(root->right, arr, top + 1);
    }
    if (!root->left && !root->right) {
        printf("%c: ", root->data);
        for (int i = 0; i < top; i++)
            printf("%d", arr[i]);
        printf("\n");
    }
}
int main() {
    int n;
    printf("Enter number of characters: ");
    scanf("%d", &n);
    char data[n];
    int freq[n];
    printf("Enter characters: ");
    for (int i = 0; i < n; i++)
        scanf(" %c", &data[i]);
    printf("Enter their frequencies: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &freq[i]);
    struct HuffmanNode* root = buildHuffmanTree(data, freq, n);
    int arr[100], top = 0;
    printf("Huffman Codes:\n");
    printCodes(root, arr, top);
    return 0;
}
```

Output:



```
Enter number of characters: 9
Enter characters: HIPOTAMUS
Enter their frequencies: 1 1 3 2 1 1 1 1 1
Huffman Codes:
M: 000
T: 001
A: 010
I: 0110
U: 0111
P: 10
H: 1100
S: 1101
O: 111

Process returned 0 (0x0)   execution time : 53.505 s
Press any key to continue.
```