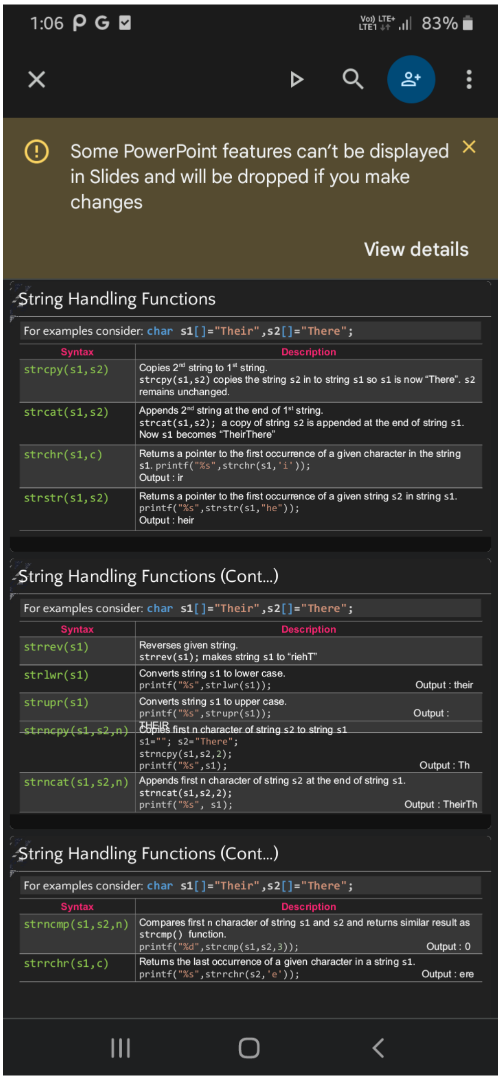
1)Enlist and explain various string handling functions.(strcat(), strlen(), strcpy(), strcmp() etc.)



Some PowerPoint features can't be displayed in Slides and will be dropped if you make changes

String Handling Functions

For examples consider: `char s1[]="Their",s2[]="There";`

Syntax	Description
`strcpy(s1,s2)`	Copies 2nd string to 1st string. `strcpy(s1,s2)` copies the string `s2` in to string `s1` so `s1` is now "There". `s2` remains unchanged.
`strcat(s1,s2)`	Appends 2nd string at the end of 1st string. `strcat(s1,s2);` a copy of string `s2` is appended at the end of string `s1`. Now `s1` becomes "TheirThere"
`strchr(s1,c)`	Returns a pointer to the first occurrence of a given character in the string `s1`. `printf("%s",strchr(s1,'i'));` Output : ir
`strstr(s1,s2)`	Returns a pointer to the first occurrence of a given string `s2` in string `s1`. `printf("%s",strstr(s1,"he"));` Output : heir

String Handling Functions (Cont...)

For examples consider: `char s1[]="Their",s2[]="There";`

Syntax	Description
`strrev(s1)`	Reverses given string. `strrev(s1);` makes string `s1` to "riehT"
`strlwr(s1)`	Converts string `s1` to lower case. `printf("%s",strlwr(s1));` Output : their
`strupr(s1)`	Converts string `s1` to upper case. `printf("%s",strupr(s1));` Output : THEIR

`strncpy(s1,s2,n)`	Copies first n character of string `s2` to string `s1`
	`s1=""; s2="There";`
	`strncpy(s1,s2,2);`
	`printf("%s",s1);` Output : Th

`strncat(s1,s2,n)`	Appends first n character of string `s2` at the end of string `s1`. `strncat(s1,s2,2);` `printf("%s", s1);` Output : TheirTh

String Handling Functions (Cont...)

For examples consider: `char s1[]="Their",s2[]="There";`

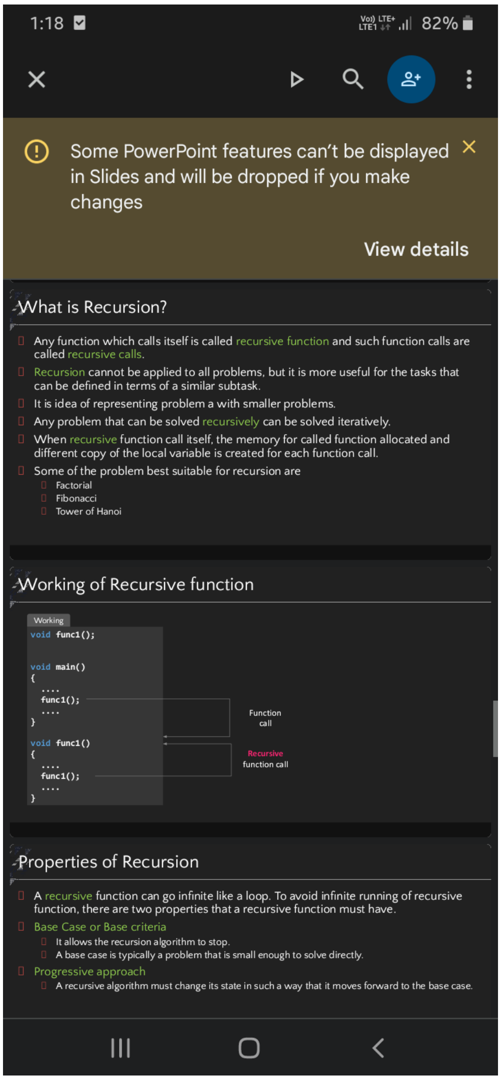
Syntax	Description
`strncmp(s1,s2,n)`	Compares first n character of string `s1` and `s2` and returns similar result as `strcmp()` function. `printf("%d",strcmp(s1,s2,3));` Output : 0
`strrchr(s1,c)`	Returns the last occurrence of a given character in a string `s1`. `printf("%s",strrchr(s2,'e'));` Output : ere

4)Describe the elements of the user defined function.

In C, a user-defined function typically consists of:

1. Return Type: Specifies the data type of the value the function returns, such as `int`, `float`, `void`, etc.
2. Function Name: The identifier for the function, following C's naming rules.
3. Parameters: Input values the function receives, enclosed in parentheses and separated by commas. Parameters are optional.
4. Function Body: The block of code enclosed in curly braces `{}` that defines the actions performed by the function.

5)Explain recursion with the help of an example.



Some PowerPoint features can't be displayed in Slides and will be dropped if you make changes

View details

What is Recursion?

- Any function which calls itself is called recursive function and such function calls are called recursive calls.
- Recursion cannot be applied to all problems, but it is more useful for the tasks that can be defined in terms of a similar subtask.
- It is idea of representing problem a with smaller problems.
- Any problem that can be solved recursively can be solved iteratively.
- When recursive function call itself, the memory for called function allocated and different copy of the local variable is created for each function call.
- Some of the problem best suitable for recursion are
  - Factorial
  - Fibonacci
  - Tower of Hanoi

Working of Recursive function

Working

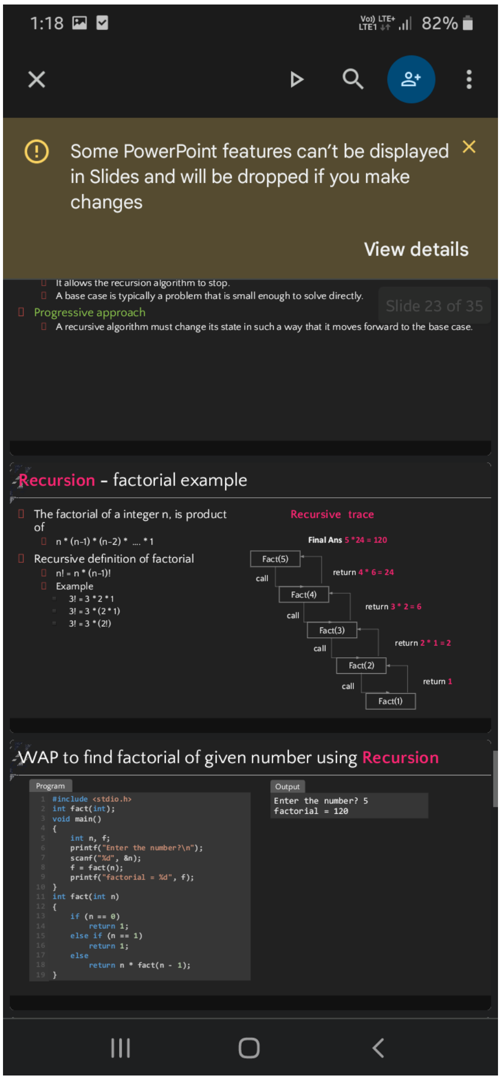
```
void func1();


void main()
{
  ....
  func1();
  ....
}

void func1()
{
  ....
  func1();
  ....
}
```

Function call

Recursive function call

Properties of Recursion

- A recursive function can go infinite like a loop. To avoid infinite running of recursive function, there are two properties that a recursive function must have.
- Base Case or Base criteria
  - It allows the recursion algorithm to stop.
  - A base case is typically a problem that is small enough to solve directly.
- Progressive approach
  - A recursive algorithm must change its state in such a way that it moves forward to the base case.



Some PowerPoint features can't be displayed in Slides and will be dropped if you make changes

- It allows the recursion algorithm to stop.
- A base case is typically a problem that is small enough to solve directly.

- Progressive approach
    - A recursive algorithm must change its state in such a way that it moves forward to the base case.

Recursion – factorial example

- The factorial of a integer n, is product of
    - n * (n-1) * (n-2) * .... * 1
- Recursive definition of factorial
    - n! = n * (n-1)!
    - Example
        - 3! = 3 * 2 * 1
        - 3! = 3 * (2 * 1)
        - 3! = 3 * (2!)

Recursive trace

Final Ans 5 *24 = 120

Fact(5)

call

return 4 * 6 = 24

Fact(4)

call

return 3 * 2 = 6

Fact(3)

call

return 2 * 1 = 2

Fact(2)

call

return 1

Fact(1)

WAP to find factorial of given number using Recursion

Program

```c
#include <stdio.h>
int fact(int);
void main()
{
    int n, f;
    printf("Enter the number?\n");
    scanf("%d", &n);
    f = fact(n);
    printf("factorial = %d", f);
}
int fact(int n)
{
    if (n == 0)
        return 1;
    else if (n == 1)
        return 1;
    else
        return n * fact(n - 1);
}
```

Output

```
Enter the number? 5
factorial = 120
```