University of Colombo School of Computing
SCS 1304 - Problem Solving Strategies and Computation Approaches

Lab Sheet 05

Tracing a Recursive Function with a Trace Table & Trace Tree

01. A bacteria colony doubles its population every hour. However, once the population reaches 2,000,000, it starts to decline by half every hour due to limited resources. Given an initial population of **P** bacteria, Use the following recursive function to calculate the population after **n** hours.

```
function bacteria_growth(P, n):
    if n == 0:
        return P
    else if P >= 2000000:
        return bacteria_growth(P / 2, n - 1)
    else:
        return bacteria_growth(P * 2, n - 1)
```

I. Create a trace table with columns for the function call, the value of **P**, the value of **n**, and the return value.

Sample table format ;

Function	P	n	Return

II. Fill in the table step-by-step as the function is called and returns values for bacteria_growth(1000, 5).

III. What is the initial population of the bacteria?

IV. What is the condition for the base case in this function?

V. Describe the recursive case when the population is less than 2,000,000.

VI. Describe the recursive case when the population is greater than or equal to 2,000,000.

VII. What value is returned when the function reaches the base case?

VIII. What is the return value for bacteria_growth(1000, 5)?

IX. Modify the function to include a limit where the population can no longer decline (e.g., a minimum population of 10). Provide the pseudo code and trace table for bacteria_growth(1000, 10, 100) where the third parameter is the minimum population limit.

2. A delivery drone starts from a warehouse and travels to deliver packages to various destinations. After each delivery, it returns to the warehouse to pick up the next package. The distance to each destination is given in an array. Use the following recursive function to calculate the total distance traveled by the drone after delivering all the packages.

```
function total_distance(drone_distance, index):
    if index == length(drone_distance):
        return 0
  else:
     return 2 * drone_distance[index] + total_distance(drone_distance, index + 1)
```

I. Trace a tree, start with the root node as total_distance([7, 12, 4], 0).

II. Draw branches for each recursive call until you reach the base case.

III. Label each node with the function call and its return value.

IV. What is the initial list of distances to the destinations?

V. What is the condition for the base case in this function?

VI. Describe the recursive case for calculating the total distance.

VII. What value is returned when the function reaches the base case?

VIII. How many recursive calls are made to calculate the total distance?

IX. What is the total distance traveled by the drone?

Quizzes

01. What type of recursion is used when solving the Tower of Hanoi problem?

A) Tail recursion

B) Head recursion

C) Tree recursion

D) Direct recursion.

02. What is a potential problem with deep recursion?

A) Memory leaks

B) Stack overflow errors.

C) Faster execution

D) Improved readability

03. In a recursive function, what must be included to avoid infinite recursion?

A) A loop

B) A counter

C) A base case.

D) A pointer

04. Which statement is true about tail recursion?

A) The recursive call is the first operation in the function

B) There are no recursive calls

C) The recursive call is the last operation in the function.

D) Tail recursion does not use stack frames

05. What is a common application of indirect recursion?

A) Optimizing memory usage

B) Problem decomposition into smaller interdependent subproblems.

C) Enhancing speed

D) Simplifying base cases

06. What can be a drawback of indirect recursion?

A) Easier debugging

B) Faster execution

C) Additional complexity due to interdependencies between functions.

D) Reduced memory usage

07. In C, how do you define a recursive function?

A) By using loops

B) By using pointers

C) By having a function call itself within its body.

D) By defining multiple functions

08. What is the result of the recursive function call factorial(5) in C?

```
int factorial(int n) {
  if (n == 0) return 1;
  else return n * factorial(n - 1);
}
```

A) 24 B) 120.

C) 720 D) 60

09. Which of the following is NOT an example of direct recursion?

A) A function that calculates factorial by calling itself

B) A function that sums numbers by calling itself

C) A function that does not call itself but calls another function in a cycle.

D) A function that prints a list by calling itself

10. What is the main benefit of tail recursion optimization?

A) It increases code complexity

B) It eliminates the need for maintaining multiple stack frames, enhancing performance.

C) It uses more memory

D) It slows down execution

11. Which of the following is an example of nested recursion?

A) A function that calls another function

B) A function that calls itself multiple times within its own definition.

C) A function that does not call itself

D) A function that uses loops

12. What does the following recursive function compute in C?

```
int sum(int n) {
   if (n == 0) return 0;
   else return n + sum(n - 1);
}
```

A) Factorial of n

B) Product of n

C) The sum of the first n natural numbers.

D) The difference of n

13. Which type of recursion is preferred for improving memory usage?

A) Head recursion B) Indirect recursion

C) Tree recursion D) Tail recursion.

14. Which statement is true about head recursion?

A) The recursive call is the first statement executed by the function.

B) The recursive call is the last statement executed by the function

C) It involves multiple functions

D) It does not use stack frames

15. Which of the following is a characteristic of direct recursion?

A) The function does not call itself

B) The function calls another function

C) The function directly calls itself.

D) The function uses iteration

16. What happens when the base case is not defined in a recursive function?

A) The program runs faster

B) The function will not execute

C) It leads to infinite recursion.

D) The function returns zero

17. Why is code modularity an advantage of indirect recursion?

A) It increases speed

B) It reduces memory usage

C) It improves readability and maintainability by dividing logic across multiple functions.

D) It simplifies the base case

18. Find the result of compute(4) for the following C code:

```
int compute(int n) {
    if (n <= 1) return n;
    return n * compute(n - 2);
}
```

A) 6 B) 8

C) 0 D) 12