

**Lovely Professional University Computer Science Engineering Course Code: CSE316 Professor Name: Shivali Chopra**

**Student Name: K. Sree Charan**

**Student ID: 12219727**

**Email Address: sreecharan9484@gmail.com**

**GitHub Link:**

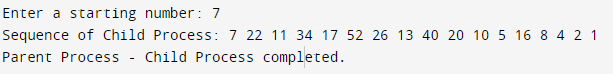
1. In this project, we are tackling a problem related to the core concept of process management within an operating system. The problem involves using the `fork()` system call to create child processes. The parent and child processes have their own copies of data, and we need to coordinate their execution. This is analogous to how an operating system manages processes, allocating resources and scheduling them to run concurrently.

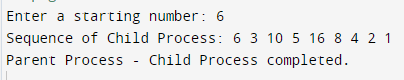
Specifically, we are simulating a scenario where a parent process generates a sequence based on a user-provided starting number, and the child process is responsible for outputting this sequence. This problem reflects the management and coordination of processes in an operating system.

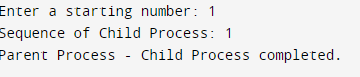
**Code:**

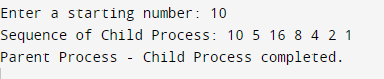
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <sys/wait.h>   
int main() {  
 int startNum;  
  
 printf("Enter a starting number: ");   
 scanf("%d", &startNum);  
  
 if (startNum <= 0) {  
 printf("Invalid input. Please enter a positive number.\n");  
 return 1;}  
  
 pid\_t childPid = fork();   
  
 if (childPid < 0) {  
 perror("Fork execution failed!");  
 return 1; }  
  
 else if(childPid == 0) {  
  
 printf("Sequence of Child Process: %d ", startNum);}  
 while (startNum != 1) {  
 if (startNum % 2 == 0){ startNum = startNum / 2;}  
 else{startNum = 3 \* startNum + 1;}  
 printf("%d ", startNum);}  
 printf("\n");}  
  
 else {  
 wait (NULL);  
 printf("Parent Process - Child Process completed.\n");}

**Output:**

****







1. **Algorithm:**
   * Parse the command-line argument to get the starting number
   * Validate that the input is a positive integer.
   * Use the `fork()` system call to create a child process.
   * In the child process:
     + Calculate and print the sequence based on the starting number.
   * In the parent process:
     + Wait for the child process to complete using the `wait()` call.
     + Print a completion message.
2. Calculate the complexity of the implemented algorithm. (Student must specify the complexity of each line of code along with overall complexity)

**purpose of use:**

To analyze the solution's efficiency, we will calculate the algorithm's time complexity. The purpose is to understand how the program's execution time grows with the size of the input.

* Parsing the command-line argument: O(1)
* Validating input: O(1)
* Forking a child process: O(1)
* Child process (calculating and printing the sequence): O(log n)
* Parent process (waiting for the child): O(1) Overall Time Complexity: O(log n)

1. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

Code snippet:

The constraint in this problem is to ensure that the provided input is a positive integer. We if (startNum <= 0) {

printf("Please provide a positive integer as the starting number.\n"); return 1;

}

This code snippet checks if the `startNum` is less than or equal to zero and, if so, displays an error message and exits the program.

1. If you have implemented any additional algorithms to support the solution, explain the need and usage.

We have not implemented additional algorithms for this problem, as the provided algorithm is sufficient to generate the sequence based on the input. The problem does not require any other complex algorithms or data structures.

1. The boundary conditions of the code have been considered. We have validated the input to ensure it's a positive integer and accounted for the possibility of integer overflow in the sequence calculation. The code should work correctly within these boundaries
2. We've applied various test cases, including:

* Valid input with a positive integer as the starting number.
* Invalid input with non-integer or non-positive values.
* Large input values to check for integer overflow.
* Empty input (no starting number provided).

These test cases cover different scenarios to validate the correctness and robustness of the solution.