Computer Programming 1:
Case Study

Output Submitted In Fulfillment

of the Requirement For

Computer Programming 1

Proponents:

Lagahit, Maenard Rafael B.

Llego, Francis Loise M.

Isidro, James Patrick M.

BSIT 1-1

Ms. Rosita Canlas

2024

Table of Contents

Case Study 1

N factorial can be defined as the product of all integers from 1 to N and it is denoted by the symbol N!. 0! (zero factorial) is defined as 1. Write a program that will input N and display N factorial. (Determine first if N is a nonnegative integer). Repeat the process as often as desired.

I. Problem Definition

Objectives

The objective of this program is to find the n factorial of the user input, which is non-negative integer, and display the process of getting the n factorial.

Output

The n factorial of the non-negative integer from the user's input, and the process of getting the n factorial.

Input

A single non-negative integer from the user.

Processing

- **Validate User Input:** Get the non-negative integer, labeled as n, from the user's input. Identify if the user entered a non-negative integer. If the user has entered anything other than a non-negative integer (negative integers, and other characters), it will be considered as an invalid input, and the program will ask the user to input a non-negative integer.
- **Calculate The Factorial:** If the input is valid, have n2 multiplied to the variable i from a for loop, and only stop multiplying until i has reached n (1 x 1, 1 x 2, 2 x 3...)

- **Log The Values:** List all of the numbers used to multiply, which is the variable i from the for loop.
- **Output:** Print the final value, which is the product of n2.
- **Ask For Repetition:** After the output has been displayed, the program will ask if the user wants to enter another integer.

II. Program Coding

Programming Language used: C language

Source Code:

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

int main() {
    int repIndex = 0; // keeping track of repetition
    printf("CASE STUDY 1: This program shows the factorial of a number from the
user input.\n");
    while (1) {
    if (repIndex >= 1) { // Input Invalidation Sequence
        printf("Invalid input. Please try again!\n");
        repIndex = 0;
    }
        int n;
        printf("\nEnter a number: ");
        if (scanf("%d", &n) != 1|| n < 0) { // get the input and check validation
at the same time
            repIndex = 1; // trigger the invalidation sequence
            while (getchar() != '\n'); // clear input buffer
            continue; // skip this iteration
        }
        while (getchar() != '\n'); // clea input buffer (doesn't affect non-
numeric characters)


            unsigned long long int n2 = 1; // to handle very large positive
integers
            printf("\n\nN = %d\n", n);
            printf("N! = ");
            int factorialOverflowCheck = 0;
```

```
            int i = 1;
            for (; i <= n; i++) { // increase i until n
                n2 *= i; // n2 = n2 * i, multiply n2 to i (1*1, 1*2, 2*3, 6*4,
24*5, ...)
                if (i == n) { // print all values used (1 x 2 x 3 x ...)
                    printf("%d", i);
                } else {
                    printf("%d x ", i);
                }
            }
            if (n >= 20) { // overflow check (>20!), if the final value exceeds
the maximum 64-bit integer
                printf("\n\nFactorial of %d exceeds the 64-bit integer limit!",
n);
                factorialOverflowCheck = 1;
            }

            if (factorialOverflowCheck != 1) {
                printf("\nN = %llu", n2); // prints the factorial number in
unsigned long long int
            }

            char repeat;
            printf("\n\nDo you want to repeat the process? Y or N "); // repeat
section
            repeat = getch(); // User enters the char without needing to wait for
the enter key
            if (repeat == 'Y' || repeat == 'y') {
                printf("\n\n");
            } else {
                printf("\n\n\nPress any key to exit...");
                getch(); // to exit the program
                break;
            }
    }
    return 0;
}
// Made by Francis Loise M. Llego from Group 17 BSIT 1-1
```

III. Program Testing

The following are sample data inputted and the result of the program:

Sample Output 1:

CASE STUDY 1: This program shows the factorial of a number from the user input.

Enter a number: 5

N = 5

N! = 1 x 2 x 3 x 4 x 5

N = 120

Do you want to repeat the process? Y or N

Sample Output 2:

CASE STUDY 1: This program shows the factorial of a number from the user input.

Enter a number: 21

N = 21

N! = 1 x 2 x 3 x 4 x 5 x 6 x 7 x 8 x 9 x 10 x 11 x 12 x 13 x 14 x 15 x 16 x 17 x 18 x 19 x 20 x 21

Factorial of 21 exceeds the 64-bit integer limit!

Do you want to repeat the process? Y or N

Sample Output 3:

CASE STUDY 1: This program shows the factorial of a number from the user input.

Enter a number: -5

Invalid input. Please try again!

Enter a number:

Sample Output 4:

CASE STUDY 1: This program shows the factorial of a number from the user input.

Enter a number: abc123

Invalid input. Please try again!

Enter a number:

Sample Output 5:

CASE STUDY 1: This program shows the factorial of a number from the user input.

Enter a number: #$%

Invalid input. Please try again!

Enter a number:

Documentation

User Documentation

These instructions are intended as a guide for the user to use this program:

1. Launch the Factorial.exe file.
2. Type a non-negative integer, and press enter to continue. Please only put one number at a time. If the input is two or more numbers, only the first number will be used and calculated. If the input is anything other than a non-negative integer, the program will discard this, and will ask to input a valid non-negative integer.
3. The program will display the process of getting the factorial of the non-negative integer and the factorial of that integer.
4. Once done, the program will ask if the user wants to repeat the process all over again. If Yes (user input is 'Y' or 'y'), then the program clears the console, and restarts the whole program once more. If No (user input is anything other than 'Y' or 'y'), then the program will exit.

Operator Documentation

The following steps will guide the operator when the program flashes an error:

1. The program uses a 64-bit integer limit for its factorial calculation. Any factorials of 20! and above exceeds this integer limit. Please refrain from using excessive positive or negative digits, as this may result in an infinite loop, or showing 0.
2. If the program shows “Press any key to continue…”, it means that it has redirected you to its exit stage. Please refrain from using a combination of integers and letters.

Case Study 2

The proper divisor of an integer N are the positive divisors less than N, a positive integer is said to be DEFICIENT, PERFECT or ABUNDANT numbers if the sum of its proper divisors is less than, equal to or greater than the number respectively. Repeat the process as often as desired.

I. Problem Definition

Objectives

The objective of the problem is to determine the proper divisors of the positive integer (N) inputted by the user (excluding the number itself), the sum of the proper divisors, and if the positive integer is either deficient, perfect, or abundant as relative to the sum of its proper divisors.

Output

The output requirements are the proper divisors of the positive integer entered by the user, the sum of the proper divisors, and its rating either as deficient, perfect, or abundant.

Input

The input requirement is a positive integer represented by N.

Processing

- **Utilize Control Structures and Operators:** To accomplish the output requirement, it is essential to use conditional statements, loops, arithmetic operators (including increment ++ and modulo %), relational operators (e.g., >, <, ==), and logical operators (e.g., &&, ||).
- **Determine Proper Divisors:** Identify the proper divisors of the given integer (excluding the integer itself).

- **Calculate Sum:** Compute the sum of the proper divisors.
- **Evaluate Status:** Compare the integer with the sum of its proper divisors to classify it as:

Perfect (equal to the sum), Abundant (less than the sum), and Deficient (greater than the sum).

II. Program Coding

Programming Language used: C language

Source Code:

```c
#include <stdio.h>


int main() {


    //Declaration of variables used in the program.
    int N;
    int divisor;
    int sum;
    int decide;

    printf("CASE STUDY 2: A program that determines the proper divisors of a number, the sum of it, and if it is deficient, perfect, or abundant.");


    //Loop used to ask the user if they want to try the program again.
    do {
    N = 0, divisor = 0, sum = 0; //Resetting the value of the variables.

    printf("\nInput a positive number: ");
    //Checks if the value inputted by the user is a positive and valid value.
    while (1) {
        if (scanf("%d", &N) == 1 && N > 0) {
            break; // Exits the while loop and heads to the processing.
            }
        else {
            printf("Invalid input! Please enter a positive number only: ");
            while (getchar() != '\n'); // Clear the input buffer.
```

```
            }
        }

    printf("Proper divisors are: ");
    //Loop that determines the proper divisors.
    for (int i = 1; i < N; i++) {

        if (N % i == 0) {
        printf("%d ", i);
        sum = sum + i;
        }

    }
     printf("\nSum of proper divisors: %d", sum);
     //Checking the value of sum relative to the value of the number inputted.
     if (sum < N) {
         printf("\n%d < %d is DEFICIENT", sum, N);
     }

     else if (sum == N) {
         printf("\n%d = %d is PERFECT", sum, N);
     }

     else if (sum > N) {
         printf("\n%d > %d is ABUNDANT", sum, N);
     }

     //Asks the user if they want to try again.
     printf("\nDo you want to do it again? Press 1 (yes) or 0 (no): ");
     scanf("%d", &decide);

     //Activates when the user inputted values other than 1 and 0.
     while (decide != 1 && decide != 0) {
         printf("Please enter 1 and 0 only. Do you want to try again? 1 (yes) or
0 (no): ");
         scanf("%d", &decide);
     }

    } while (decide == 1);

    //Message that indicates program termination.
    printf("Program terminated.");

    return 0;
}
```

```
// Programmed and Designed by James Patrick M. Isidro from Group 17 BSIT 1-1
```

III. Program Testing

The following are sample data inputted and the result of the program:

Sample Output 1:

CASE STUDY 2: A program that determines the proper divisors of a number, the sum of it, and if it is deficient, perfect, or abundant.

Input a positive number: 10

Proper divisors are: 1 2 5

Sum of proper divisors: 8

8 < 10 is DEFICIENT

Do you want to do it again? Press 1 (yes) or 0 (no):

Sample Output 2:

CASE STUDY 2: A program that determines the proper divisors of a number, the sum of it, and if it is deficient, perfect, or abundant.

Input a positive number: 28

Proper divisors are: 1 2 4 7 14

Sum of proper divisors: 28

28 = 28 is PERFECT

Do you want to do it again? Press 1 (yes) or 0 (no):

Sample Output 3:

CASE STUDY 2: A program that determines the proper divisors of a number, the sum of it, and if it is deficient, perfect, or abundant.

Input a positive number: 24

Proper divisors are: 1 2 3 4 6 8 12

Sum of proper divisors: 36

36 > 24 is ABUNDANT

Do you want to do it again? Press 1 (yes) or 0 (no):

Sample Output 4:

CASE STUDY 2: A program that determines the proper divisors of a number, the sum of it, and if it is deficient, perfect, or abundant.

Input a positive number: -24

Invalid input! Please enter a positive number only:

Sample Output 5

CASE STUDY 2: A program that determines the proper divisors of a number, the sum of it, and if it is deficient, perfect, or abundant.

Input a positive number: @

Invalid input! Please enter a positive number only:

IV. Documentation

User Documentation

The following steps will guide the users in running and testing the program:

1. Copy and paste the code to a compiler for C language.

2. Save the file as a C program file.
3. Click the run button for the program to execute.
4. Input only a positive integer and press ‘enter’ to acquire the results.
5. Press number ‘1’ if the user wants to try the program again or press ‘0’ for the program to terminate.

Operator Documentation

The following steps will guide the operator when the program flashes an error:

1. Refrain from inputting negative numbers and symbols.
2. Avoid entering excessively large positive or negative numbers, as this may lead to an infinite loop or a result of 0.

Case Study 3

Write a program that will input a non-negative integer and determine if the integer is DWARF or NOT. An integer is said to be DWARF if the sum of its factors is greater than the half of the number. Repeat the process as often as desired.

I. Problem Definition:

Objectives

The objective of the program is to determine if a non-negative integer is "DWARF" by checking if the sum of its factors exceeds half the number and to allow the user to repeat this process for multiple inputs

Output

The factors of the non-negative integer that the user gave, and if the integer is a Dwarf or Not.

Input

Input a non-negative number from the user.

Processing

- **Validate the integer** : Get the non-negative integer, labeled as number, from the user's input. Identify if the user entered a non-negative integer by checking if the number is greater than zero. If the user inputs a negative number,it will be considered as an invalid input, and the program will ask the user to input a valid integer.
- **Find the factors :** Get the factors of the user’s integer by using the for loop and finding all numbers that divide the user's integer evenly.
- **Sum of the factors** : Add all the factors of the user's integer.

- **Dwarf or not :** If the half of the user's integer is greater than the sum of the factors the result is a Dwarf, otherwise the integer is not a Dwarf.

II. Program Coding

Programming Language used: C language

Source Code:

```c
#include <stdio.h>

int main() {
    int number, i, sum, repeat = 1;

    printf("CASE STUDY 3: A program that determines the factors of a positive number, the  sum of the factors, and if it is Dwarf or Not Dwarf.\n\n");

    while (repeat) {
        printf("Enter a non-negative integer ");
        scanf("%d", &number);

        if (number < 0) {
            printf("Please enter a non-negative integer!\n");
            continue;
        }

        printf("Factors are: ");
        sum = 0;
        for (i = 1; i <= number / 2; i++) {
            if (number % i == 0) {
                printf("%d ", i);
                sum += i;
            }
        }

        printf("\nSum of its factors: %d\n", sum);
        printf("Half of the number: %.2f\n", number / 2.0);


        if (sum > number / 2.0) {
            printf("%d is DWARF\n", number);
        } else {
            printf("%d is NOT DWARF\n", number);
```

```
        }


        printf("\nDo you want to enter another number? (1 for Yes, 0 for No): ");
        scanf("%d", &repeat);
    }

    return 0;
}
```

III. Program Testing

The following are sample data inputted and the result of the program:

Sample output 1:

CASE STUDY 3: A program that determines the factors of a positive number, the sum of the factors, and if it is Dwarf or Not Dwarf.

Enter a non-negative integer (or -1 to exit): 15

Factors are: 1 3 5

Sum of its factors: 9

Half of the number: 7.50

15 is DWARF

Do you want to enter another number? (1 for Yes, 0 for No) :

Sample output 2:

CASE STUDY 3: A program that determines the factors of a positive number, the sum of the factors, and if it is Dwarf or Not Dwarf.

Enter a non-negative integer (or -1 to exit): 8

Factors are: 1 2 4

Sum of its factors: 7

Half of the number: 4.00

8 is DWARF

Do you want to enter another number? (1 for Yes, 0 for No):

Sample output 3:

CASE STUDY 3: A program that determines the factors of a positive number, the sum of the factors, and if it is Dwarf or Not Dwarf.

Enter a non-negative integer (or -1 to exit): 20

Factors are: 1 2 4 5 10

Sum of its factors: 22

Half of the number: 10.00

20 is DWARF

Do you want to enter another number? (1 for Yes, 0 for No):

IV. Documentation

User Documentation

The following steps will guide the users in running and testing the program:

1. Launch the C file.
2. Input or type a single non-negative number. If the input is two or more numbers, only the first number will be used and calculated. If the input is anything other than a non-negative integer, the program will discard this, and terminate the program.
3. The program will display the factors of the given integer, and if that integer is a Dwarf or not.
4. Once done, the program will ask if the user wants to repeat the process all over again. Press number '1' to try the program again or press '0' for the program to terminate.

Operator Documentation

The following steps will guide the operator when the program flashes an error:

1. Please refrain from using or inputting negative numbers, symbols or letters to avoid unexpected termination of the program.
2. Use -1 to terminate the program and stop it from starting the loop.