भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी

Indian Institute of Information Technology Guwahati

COMPUTER PROGRAMMING LAB (CS110) ASSIGNMENTS AND SOLUTIONS–06

[Note: Do not use the `scanf()` function.]

1. Write a function in C to compute the area of a circle. You must pass the circle's diameter as a parameter to the function. Write the main function and call it from the main function. You need to define the function before the main function.

```c
#include <stdio.h>
#include <math.h>

double area(double diameter) {
    return M_PI * diameter * diameter / 4.0;
}

int main() {
    double d = 2.0;
    printf("Area: %g\n", area(d));
    return 0;
}
```

2. Write a function in C to check whether a number is prime or not. If it is a prime number, the function should return 1. Otherwise, it should return 0. Write the main function and call the function from the main function. You need to define the function after the main function and declare the function inside the main function.

```c
#include <stdio.h>
#include <math.h>

int main() {
    int n = 37, isPrime(int n);
    printf(isPrime(n) ? "Prime\n" : "Not Prime\n");
    return 0;
}

int isPrime(int n) {
    double sqrt_n = sqrt(n);
    for (int i = 2; i <= sqrt_n; i++)
```

```c
        if (n % i == 0)
            return 0;
    return 1;
}
```

3. Write a function in C to check whether a number is Armstrong or not. If it is an Armstrong number, the function should return 1. Otherwise, it should return 0. Write the main function and call the function from the main function. You need to define the function after the main function and declare the function outside (before) the main function.

```c
#include <stdio.h>
#include <math.h>

int isArmstrong(int n);

int main() {
    int n = 371;
    printf(isArmstrong(n) ? "Armstrong\n" : "Not Armstrong\n");
    return 0;
}

int isArmstrong(int n) {
    int digits = 0, sum = 0;
    for (int m = n; m; m /= 10)
        digits++;
    for (int m = n; m; m /= 10)
        sum += (int) pow(m % 10, digits);
    return n == sum;
}
```

4. Write a function in C to print all prime numbers inside a given range $[a, b]$. You need to pass a and b as parameters to the function.

```c
#include <stdio.h>
#include <math.h>

int isPrime(int n) {
    double sqrt_n = sqrt(n);
    for (int i = 2; i <= sqrt_n; i++)
        if (n % i == 0)
            return 0;
    return 1;
}

void printPrimes(int a, int b) {
    for (int n = a; n <= b; n++)
        if (isPrime(n))
            printf("%d\n", n);
}

int main() {
    int a = 30, b = 300;
    printPrimes(a, b);
    return 0;
}
```

5. Write a function in C to print all Strong numbers inside a given range $[a, b]$. You need to pass a and b as parameters to the function.

```c
#include <stdio.h>

int factorial(int n) {
    int value = 1;
    for (; n > 0; n--)
        value *= n;
    return value;
}

int isStrong(int n) {
    int sum = 0;
    for (int m = n; m; m /= 10)
        sum += factorial(m % 10);
    return n == sum;
}

void printStrongNumbers(int a, int b) {
    for (int n = a; n <= b; n++)
        if (isStrong(n))
            printf("%d\n", n);
}

int main() {
    int a = 10, b = 50000;
    printStrongNumbers(a, b);
    return 0;
}
```

6. Write a function in C to print all Armstrong numbers inside a given range $[a, b]$. You need to pass a and b as parameters to the function.

```c
#include <stdio.h>
#include <math.h>

int isArmstrong(int n) {
    int digits = 0, sum = 0;
    for (int m = n; m; m /= 10)
        digits++;
    for (int m = n; m; m /= 10)
```

```c
        sum += (int) pow(m % 10, digits);
    return n == sum;
}

void printArmstrongNumbers(int a, int b) {
    for (int n = a; n <= b; n++)
        if (isArmstrong(n))
            printf("%d\n", n);
}

int main() {
    int a = 10, b = 3000;
    printArmstrongNumbers(a, b);
    return 0;
}
```

7. Write a function in C to print all Perfect numbers inside a given range $[a, b]$. You need to pass a and b as parameters to the function.

```c
#include <stdio.h>

int isPerfect(int n) {
    int n_2 = n / 2, sum = 0;
    for (int i = 1; i <= n_2; i++)
        if (n % i == 0)
            sum += i;
    return n == sum;
}

void printPerfectNumbers(int a, int b) {
    for (int n = a; n <= b; n++)
        if (isPerfect(n))
            printf("%d\n", n);
}

int main() {
    int a = 10, b = 30000;
    printPerfectNumbers(a, b);
    return 0;
}
```

8. Write a function in C to find the minimum of two numbers.

```c
#include <stdio.h>

int max(int a, int b) {
    return a > b ? a : b;
}

int main() {
    int a = 2, b = 3;
    printf("%d", max(a, b));
    return 0;
}
```

9. Write a function in C that takes three integers as arguments and returns the largest one's value.

```c
#include <stdio.h>

int max(int a, int b, int c) {
    return a > b ? (a > c ? a : c)
                 : (b > c ? b : c);
}

int main() {
    int a = 4, b = 2, c = 1;
    printf("%d", max(a, b, c));
    return 0;
}
```

10. Write a function in C that takes a real number as an argument and returns that number's absolute value.

```c
#include <stdio.h>

double absolute(double real) {
    return real > 0.0 ? real : - real;
}

int main() {
    double real = -2.3;
    printf("%lg", absolute(real));
    return 0;
}
```

11. Write a function in C that takes a positive integer n as an argument and returns the smallest power of two that is greater than or equal to n.

```c
#include <stdio.h>

int smallest_power_of_2(int n) {
    int i = 1;
    while (i < n) i *= 2;
    return i;
}

int main() {
    int n = 16;
    printf("%d", smallest_power_of_2(n));
    return 0;
}
```

12. Write a function in C that takes a positive integer as input and returns the leading digit in its decimal representation. For example, the leading digit of 234567 is 2.

```
#include <stdio.h>

int leadingDigit(int n) {
    while (n > 9)
        n /= 10;
    return n;
}

int main() {
    int n = 712345;
    printf("%d", leadingDigit(n));
    return 0;
}
```

13. Write a recursive function in C to find the factorial of a positive integer.

```
#include <stdio.h>

int factorial(int n) {
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}

int main() {
    int n = 5;
    printf("%d", factorial(n));
    return 0;
}
```

14. Write a recursive function in C to find the summation of the first n natural numbers.

```
#include <stdio.h>

unsigned int factorial(unsigned int n) {
    register unsigned int sum = 0;
    for (register unsigned int i = 1; i <= n; i++)
        sum += i;
    return sum;
}

int main() {
    unsigned int n = 10;
    printf("%u", factorial(n));
```

```c
    return 0;
}
```

15. Write a recursive function in C to find the *i*th number in the Fibonacci sequence.

```c
#include <stdio.h>

int fibonacci(int i) {
    if (i == 0) return 0;
    if (i == 1) return 1;
    return fibonacci (i - 1) + fibonacci (i - 2);

}

int main() {
    int i = 10;
    printf("%d", fibonacci(i));
    return 0;
}
```

16. Write a recursive function in C to find x^n, where x is a real value, and n is a positive integer.

```c
#include <stdio.h>

double power(double x, int n) {
    if (n == 0) return 1;
    return x * power(x, n - 1);
}

int main() {
    double x = 2;
    int n = 5;
    printf("%g", power(x, n));
    return 0;
}
```

17. Write a recursive function in C to find the sum of all even numbers in a given range.

```c
#include <stdio.h>

int sumEven(int a, int b) {
    if (a > b) return 0;
    if (a % 2) return sumEven(a + 1, b);
    return a + sumEven(a + 2, b);
}

int main() {
    int a = 7, b = 13;
    printf("%d", sumEven(a, b));
    return 0;
}
```

18. Write a recursive function in C to find the sum of all odd numbers in a given range.

```
#include <stdio.h>

int sumOdd(int a, int b) {
    if (a > b) return 0;
    if (a % 2 == 0) return sumOdd(a + 1, b);
    return a + sumOdd(a + 2, b);
}

int main() {
    int a = 8, b = 11;
    printf("%d", sumOdd(a, b));
    return 0;
}
```

19. Write a recursive function in C to find the number of digits in a positive integer.

```
#include <stdio.h>

int digits(int n) {
    if (n == 0) return 0;
    return 1 + digits(n / 10);
}

int main() {
    int n = 12345;
    printf("%d", digits(n));
    return 0;
}
```

20. Write a recursive function in C to find the sum of digits in a positive integer.

```
#include <stdio.h>

int sumOfDigits(int n) {
    if (n == 0) return 0;
    return n % 10 + sumOfDigits(n / 10);
}

int main() {
    int n = 12345;
    printf("%d", sumOfDigits(n));
    return 0;
}
```

21. Write a recursive function in C to reverse a positive integer.

```c
#include <stdio.h>
#include <math.h>

int reverse(int n) {
    if (n < 10) return n;
    return (n % 10) * pow(10, (int)log10(n)) + reverse(n / 10);
}

int main() {
    int n = 12345;
    printf("%d", reverse(n));
    return 0;
}
```

```c
#include <stdio.h>

int reverse(int n, int r) {
    if (n < 10) return r * 10 + n;
    return reverse(n / 10, r * 10 + n  % 10);
}

int main() {
    int n = 12345;
    printf("%d", reverse(n, 0));
    return 0;
}
```

22. Write a recursive function in C to find the greatest common divisor (GCD) of two numbers.

```c
#include <stdio.h>

int gcd(int a, int b) {
    if (b != 0) return gcd(b, a % b);
    return a;
}

int main() {
    int a = 8, b = 12;
    printf("%d", gcd(a, b));
    return 0;
}
```

23. Write a tail-recursive function in C to find the factorial of a positive integer.

```c
#include <stdio.h>

int factorial(int n, int result) {
    if (n == 0) return result;
    return factorial(n - 1, n * result);
}

int main() {
    int n = 5;
    printf("%d", factorial (n, 1));
    return 0;
}
```

24. Write a tail-recursive function in C to find the summation of the first n natural numbers.

```
#include <stdio.h>

int sum(int n, int result) {
    if (n == 0) return result;
    return sum(n - 1, n + result);
}

int main() {
    int n = 5;
    printf("%d", sum(n, 0));
    return 0;
}
```