C Programs for Practice (Functions)

1. **C Program to determine Prime Number using function**

```
// C Program to Check Prime Number using Square Root Optimized
// Trial Division Approach
#include <math.h>
#include <stdio.h>

int isPrime(int N) {

    // Check divisibility from 2 to sqrt(N)
    for (int i = 2; i <= sqrt(N); i++) {

        // If N is divisible by i, it is not a prime number
        if (N % i == 0) {
            return 0;
        }
    }

    // If no divisors were found, N is a prime number
    return 1;
}

int main() {
    int N = 10;
      printf("Is %d prime?\n", N);

    // Check if the number is prime
    if (isPrime(N)) {
        printf("Yes\n");
    }
    else {
        printf("No\n");
    }

    return 0;
}
```

2. **C Program to determine the roots of a Quadratic Equation**

```
// C program to find roots of
// a quadratic equation
#include <math.h>
#include <stdio.h>
```

```
#include <stdlib.h>

// Prints roots of quadratic
// equation ax*2 + bx + x
void findRoots(int a, int b, int c)
{
	// If a is 0, then equation is
	// not quadratic, but linear
	if (a == 0) {
		printf("Invalid");
		return;
	}

	int d = b * b - 4 * a * c;
	double sqrt_val = sqrt(abs(d));

	if (d > 0) {
		printf("Roots are real and different\n");
		printf("%f\n%f", (double)(-b + sqrt_val) / (2 * a),
			(double)(-b - sqrt_val) / (2 * a));
	}
	else if (d == 0) {
		printf("Roots are real and same\n");
		printf("%f", -(double)b / (2 * a));
	}
	else // d < 0
	{
		printf("Roots are complex\n");
		printf("%f + i%f\n%f - i%f", -(double)b / (2 * a),
			sqrt_val / (2 * a), -(double)b / (2 * a),
			sqrt_val / (2 * a));
	}
}

// Driver code
int main()
{
	int a = 1, b = -7, c = 12;

	// Function call
	findRoots(a, b, c);
	return 0;
}
```

3. **C Program to calculate the power of a number**

```
// C program to calculate the power of a number
#include <stdio.h>

// Naive iterative solution to
// calculate pow(x, n)
long power(int x, unsigned n)
{
	// Initialize result to 1
	long long pow = 1;

	// Multiply x for n times
	for (int i = 0; i < n; i++) {
		pow = pow * x;
	}

	return pow;
}

// Driver code
int main(void)
{

	int x = 2;
	unsigned n = 3;

	// Function call
	int result = power(x, n);
	printf("%d", result);

	return 0;
}
```

4. **C Program to determine the sum of natural numbers using Recursion**

```
// C program to find the sum of n
// natural numbers using recursion
#include <stdio.h>

// Returns the sum of first n
// natural numbers
int recSum(int n)
{
// Base condition
```

```
        if (n <= 1)
                return n;

// Recursive call
        return n + recSum(n - 1);
}

// Driver code
int main()
{
        int n = 10;
        printf("Sum = %d ", recSum(n));
        return 0;
}
```

5. **C Program to determine the factorial of a number using Recursion**

```
// C program to find factorial of given number
// using recursion
#include <stdio.h>

unsigned int factorial(unsigned int n) {

   // Base Case:
   if (n == 1) {
      return 1;
   }

   // Multiplying the current N with the previous product
   // of Ns
   return n * factorial(n - 1);
}

int main() {
   int num = 5;
   printf("Factorial of %d is %d", num, factorial(num));
   return 0;
}
```

6. **C Program to calculate the GCD/HCF of two numbers**

```
// C program to find GCD of two numbers
#include <math.h>
#include <stdio.h>

// Function to return gcd of a and b
```

```
int gcd(int a, int b)
{
    // Find Minimum of a and b
    int result = ((a < b) ? a : b);
    while (result > 0) {
        // Check if both a and b are divisible by result
        if (a % result == 0 && b % result == 0) {
            break;
        }
        result--;
    }
    // return gcd of a nd b
    return result;
}

// Driver program to test above function
int main()
{
    int a = 98, b = 56;
    printf("GCD of %d and %d is %d ", a, b, gcd(a, b));
    return 0;
}
```