Q1

```
//Armstrong Number

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

int isArmstrong(int number) {
   int originalNumber = number;
   int sum = 0;
   int numDigits = 0;

   while (originalNumber != 0) {
      originalNumber /= 10;
      numDigits++;
   }

   originalNumber = number;

   while (originalNumber != 0) {
      int digit = originalNumber % 10;
      sum += pow(digit, numDigits);
      originalNumber /= 10;
   }

   return sum == number;
}

int main() {
   int number;
```

```
    printf("Enter a number: ");
    scanf("%d", &number);

    if (isArmstrong(number)) {
        printf("%d is an Armstrong number.\n", number);
    } else {
        printf("%d is not an Armstrong number.\n", number);
    }

    return 0;
}
```

Q2

```
//finding HCF

#include <stdio.h>

int calculateHCF(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main() {
    int num1, num2, hcf;
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    hcf = calculateHCF(num1, num2);

    printf("HCF of %d and %d is %d\n", num1, num2, hcf);

    return 0;
}
```

Q3

```
//Subtraction without Operators

#include <stdio.h>

int subtract(int a, int b) {
    while (b != 0) {
        int borrow = (~a) & b;

        a = a ^ b;
        b = borrow << 1;
    }
    return a;
}

int main() {
    int num1, num2, result;
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    result = subtract(num1, num2);

    printf("Result of %d - %d is %d\n", num1, num2, result);

    return 0;
}
```

Q4

```
//Swapping Integer

#include <stdio.h>

// Method 1: Using a temporary variable
void swap_with_temp(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Method 2: Using arithmetic operations
void swap_with_arithmetic(int *a, int *b) {
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
```

```
}

// Method 3: Using bitwise XOR
void swap_with_xor(int *a, int *b) {
    *a = *a ^ *b;
    *b = *a ^ *b;
    *a = *a ^ *b;
}

// Method 4: Using pointers
void swap_with_pointers(int *a, int *b) {
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}

int main() {
    int a, b;

    // Accepting two integer numbers
    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);

    printf("Original values: a = %d, b = %d\n", a, b);

    // Call Method 1
    swap_with_temp(&a, &b);
    printf("After swap_with_temp: a = %d, b = %d\n", a, b);

    // Reset values for next method
    swap_with_temp(&a, &b); // Swap back to original
    // Call Method 2
    swap_with_arithmetic(&a, &b);
    printf("After swap_with_arithmetic: a = %d, b = %d\n", a, b);

    // Reset values for next method
    swap_with_arithmetic(&a, &b); // Swap back to original

    // Call Method 3
    swap_with_xor(&a, &b);
    printf("After swap_with_xor: a = %d, b = %d\n", a, b);

    // Reset values for next method
    swap_with_xor(&a, &b); // Swap back to original

    // Call Method 4
    swap_with_pointers(&a, &b);
    printf("After swap_with_pointers: a = %d, b = %d\n", a, b);

    return 0;
}
```

Q5

```
//perfect Number

#include <stdio.h>

int isPerfectNumber(int n) {
    int sum = 0;
    for (int i = 1; i <= n / 2; i++) {
        if (n % i == 0) {
            sum += i;
        }
    }
    return (sum == n);
}
int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (isPerfectNumber(num)) {
        printf("%d is a Perfect Number.\n", num);
    } else {
        printf("%d is not a Perfect Number.\n", num);
    }

    return 0;
}
```

Q6

```
//Cordinates and Quadrant

#include <stdio.h>
```

```
void findQuadrant(int x, int y) {
    if (x > 0 && y > 0) {
        printf("The coordinate point (%d,%d) lies in the First quadrant\n", x, y);
    }
    else if (x < 0 && y > 0) {
        printf("The coordinate point (%d,%d) lies in the Second quadrant\n", x, y);
    }
    else if (x < 0 && y < 0) {
        printf("The coordinate point (%d,%d) lies in the Third quadrant\n", x, y);
    }
    else if (x > 0 && y < 0) {
        printf("The coordinate point (%d,%d) lies in the Fourth quadrant\n", x, y);
    }
    else if (x == 0 && y != 0) {
        printf("The coordinate point (%d,%d) lies on the Y-axis\n", x, y);
    }
    else if (y == 0 && x != 0) {
        printf("The coordinate point (%d,%d) lies on the X-axis\n", x, y);
    }
    else if (x == 0 && y == 0) {
        printf("The coordinate point (%d,%d) lies at the Origin\n", x, y);
    }
}

int main() {
    int x, y;

    printf("Enter the coordinates (x y): ");
    scanf("%d %d", &x, &y);

    findQuadrant(x, y);
    return 0;
}
```

Q7

```
//Binary to decimal and Decimal to binary

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

int binaryToDecimal(long long n);
long long decimalToBinary(int n);

int main() {
    int choice;
    printf("Choose an option:\n");
    printf("1. Binary to Decimal\n");
    printf("2. Decimal to Binary\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        long long binary;
        printf("Enter a binary number: ");
        scanf("%lld", &binary);
        printf("Decimal: %d\n", binaryToDecimal(binary));
    }
    else if (choice == 2) {
        int decimal;
        printf("Enter a decimal number: ");
        scanf("%d", &decimal);
        printf("Binary: %lld\n", decimalToBinary(decimal));
    }
    else {
        printf("Invalid choice.\n");
    }

    return 0;
}

int binaryToDecimal(long long n) {
    int decimal = 0, i = 0, remainder;
    while (n != 0) {
        remainder = n % 10;
        n /= 10;
        decimal += remainder * pow(2, i);
        ++i;
    }
    return decimal;
}

long long decimalToBinary(int n) {
    long long binary = 0;
    int remainder, i = 1;
    while (n != 0) {
        remainder = n % 2;
        n /= 2;
        binary += remainder * i;
        i *= 10;
    }
    return binary;
}
```

Q7

```
// Pattern

#include <stdio.h>

int main() {
    int rows, i, j;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = 1; i <= rows; i++) {
        for (j = 1; j <= i; j++) {
            if ((i + j) % 2 == 0)
                printf("1");
            else
                printf("0");
        }
        printf("\n");
    }

    return 0;
}
```

Q8

```
// Pattern

#include <stdio.h>

int main() {
    int rows, i, j;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = 1; i <= rows; i++) {
        for (j = 1; j <= i; j++) {
            if ((i + j) % 2 == 0)
                printf("1");
            else
                printf("0");
        }
        printf("\n");
    }

    return 0;
}
```

Q9

```
//Pattern

#include <stdio.h>

int main() {
    int n, i, j, k;

    printf("Enter the number of rows: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        for (j = 1; j <= i; j++) {
            if (j % 2 == 0)
                printf("1");
            else
                printf("0");
        }

        for (k = 1; k <= 2 * (n - i); k++) {
            printf(" ");
        }

        for (j = 1; j <= i; j++) {
            if (j % 2 == 0)
                printf("1");
            else
                printf("0");
        }

        printf("\n");
    }
    return 0;
}
```

Q10

```
//Pascal'c Triangle

#include <stdio.h>

int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}

int combination(int n, int k) {
    return factorial(n) / (factorial(k) * factorial(n - k));
}

int main() {
    int rows;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (int i = 0; i < rows; i++) {
        // Print spaces for alignment
        for (int j = 0; j < rows - i - 1; j++) {
            printf(" ");
        }

        // Print values in Pascal's Triangle row
        for (int k = 0; k <= i; k++) {
            printf("%d ", combination(i, k));
        }

        printf("\n");
    }

    return 0;
}
```