PROGRAM 1: Program to calculate sum, difference, product, quotient and reminder of two numbers

```c
/* Program to calculate sum, difference, product, quotient
 * and reminder of two numbers
 */

# include <stdio.h>

int main()
{
    int num_1 = 0,
        num_2 = 0,
        sum = 0,
        difference = 0,
        product = 0,
        quotient = 0,
        remainder = 0;

    printf ("Enter num_1 = ");
    scanf ("%d", &num_1);
    printf ("Enter num_2 = ");
    scanf ("%d", &num_2);

    sum        = num_1 + num_2;
    difference = num_1 - num_2;
    product    = num_1 * num_2;
    quotient   = num_1 / num_2;
    remainder  = num_1 % num_2;

    printf ("\n");
    printf ("Sum         = %d\n", sum);
    printf ("Difference  = %d\n", difference);
    printf ("Product     = %d\n", product);
    printf ("Quotient    = %d\n", quotient);
    printf ("Remainder   = %d\n", remainder);
    return 0;
}
```

```
~/C  main ±  xecut operators.c
Enter num_1 = 25
Enter num_2 = 12

Sum         = 37
Difference  = 13
Product     = 300
Quotient    = 2
Remainder   = 1
~/C  main ±  xecut operators.c
Enter num_1 = 56
Enter num_2 = 4

Sum         = 60
Difference  = 52
Product     = 224
Quotient    = 14
Remainder   = 0
~/C  main ±
```

PROGRAM 2: Program to demonstrate pre and post increment operators in C

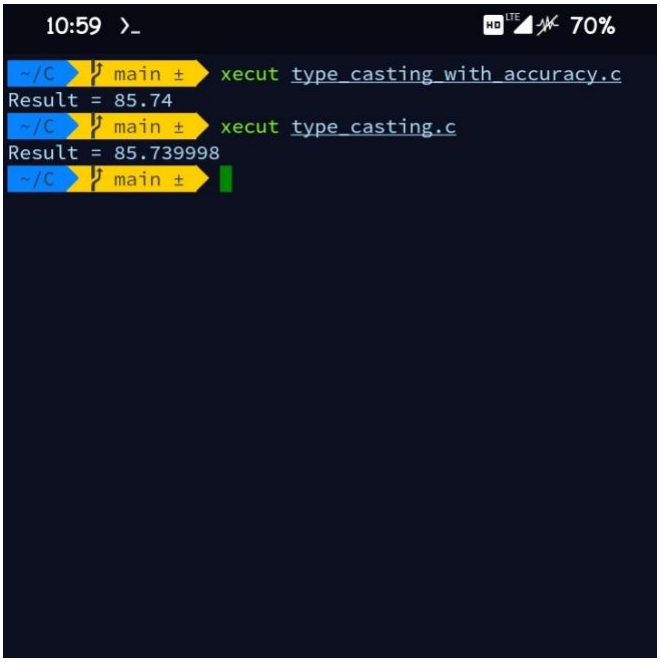
```c
/* Program to demonstrate pre-post increment operators
 */

# include <stdio.h>

int main()
{
    int x = 5, y = 7, result = 0;

    printf ("Before calculation:\n"
            "   x = %d\n"
            "   y = %d\n", x, y);

    // at this stage result = 5 + 8 and x = 6, y = 8
    result = x++ + ++y;

    printf ("\nResult = %d\n\n", result);

    printf ("After calculation:\n"
            "   x = %d\n"
            "   y = %d\n", x, y);

    return 0;
}
```

```
11:10 >_                                   69%
~/C  main ±  xecut pre_post_inc.c
Before calculation:
   x = 5
   y = 7

Result = 13

After calculation:
   x = 6
   y = 8
~/C  main ±
```

PROGRAM 3: Program to demonstrate type casting in C

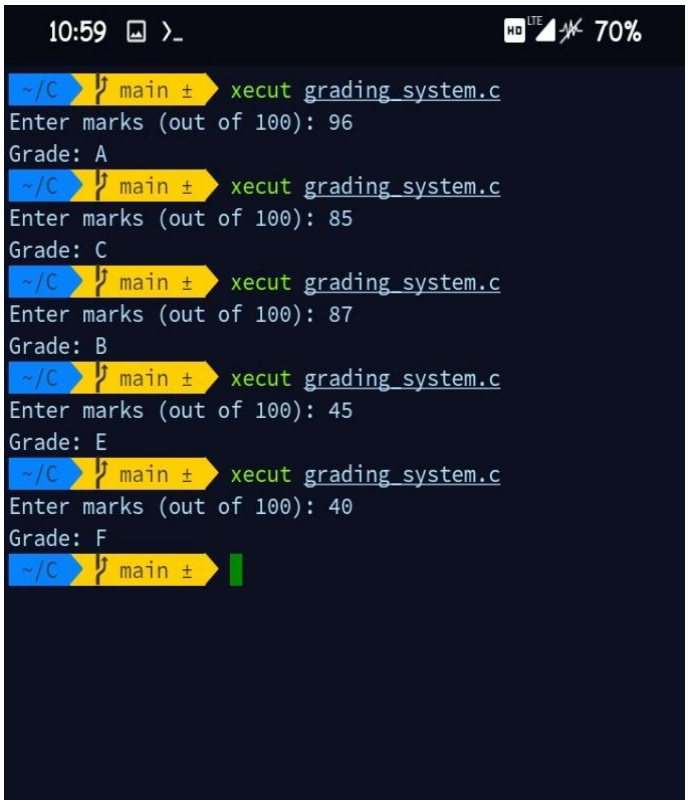
```c
/* Program to demonstrate type casting
 */

# include <stdio.h>

int main()
{
    int x = 8574, y = 100;

    /* NOTE: this casts x to float and then does the division
     * If (float) isn't used, the result will only be 85.
     */
    float result = (float) x / y;

    printf ("Result = %lf\n", result);
    return 0;
}
```

```c
/* Program to demonstrate type casting using accuracy format string
 */

# include <stdio.h>

int main()
{
    int x = 8574, y = 100;

    /* NOTE: this casts x to float and then does the division
     * If (float) isn't used, the result will only be 85.
     */
    float result = (float) x / y;

    printf ("Result = %0.2lf\n", result);
    return 0;
}
```



PROGRAM 4: WACP for grading marks obtained by a student using conditional statements

```c
/* Program for grading marks obtained by a student
 * using conditional statements.
 */

# include <stdio.h>

int main()
{
    int student_marks = 0;

    printf ("Enter marks (out of 100): ");
    scanf ("%d", &student_marks);

    if (student_marks < 0) {
        printf ("error: marks entered subceeds 0\n");
        return 0;
    } else if (student_marks > 100) {
        printf ("error: marks entered exceeds 100\n");
        return 0;
    }

    printf ("Grade: ");
    if (student_marks <= 40) {
        printf ("F\n");
    } else if (student_marks > 40 && student_marks <= 55) {
        printf ("E\n");
    } else if (student_marks > 55 && student_marks <= 70) {
        printf ("D\n");
    } else if (student_marks > 70 && student_marks <= 85) {
        printf ("C\n");
    } else if (student_marks > 85 && student_marks <= 95) {
        printf ("B\n");
    } else if (student_marks > 95 && student_marks <= 100) {
        printf ("A\n");
    }
    return 0;
}
```



EXECUTION BASH SCRIPT (xecut):

```
#!/bin/bash

file="$1"
if ! [ "$file" = "" ]
then
    exec_name=$(basename "$file" ".c")
    gcc -g -Wall "$file" -o "$exec_name"
    ./"$exec_name"
    rm "$exec_name"
else
    echo "error: no file name passed"
fi

```