```c


/*######################## EX 1 ##########################*/

/* addition program
 * it uses 2 integer numbers and prints their sum
 * also, it uses pre and post increment to demonstrate their use
 * */

#include<stdio.h>

int main(){
   int number1, number2, sum, sum2, sum3;   /* declaration of variables */

   // initialize the 2 integers
   number1=10;
   number2=20;

   sum=number1+number2;   /* assignment of sum */


   sum2 = ++number1;     /* first, number is incremented then assigne to sum2 */
   sum3 = number2++;     /* first, number is assigned to sum2 then incremented*/

   printf("Sum values are %d,  %d,  %d\n", sum, sum2, sum3); /* the sum values are printed */
   printf("Numbers are :%3d,%2d\n", number1, number2);  /* numbers are printed */

   return 0; /* return from main (to operating system)*/

}



/*######################## EX 2 ##########################*/


/* example of overflowing
 * note that the program uses char (signed)data type for the first variable
 * and unsinged char data type for the second variable
 */
#include<stdio.h>

int main(){
    char c1=193;  // first char variable is initialized to -63, as it is greater than 127 and the counter is reset from -128
    unsigned char c2=-160; // second charvariable is initialized to 96 as it is smaller than -128 and the counter continues backwards from 127
    printf("%d  %d",c1,c2); // display the numerical values for the 2 variables
    printf("\n"); // go to next line
    return 0;  // return from main
}

/*######################## EX 3 ##########################*/

/* demonstrates the use of numeric data type with different precisions
 * the "same" computations are done for both data types (float and double)
 * the results obtained are different
 */

#include <stdio.h>

int main()
{
    int i;

    float a = 1.f / 81;
    float b = 0;
    for (i = 0; i < 729; ++ i)
            b += a;
    printf("%.7g\n", b);   // prints 9.000023


    double aa = 1.0 / 81;
    double bb = 0;
    for ( i = 0; i < 729; ++ i)
            bb += aa;
    printf("%.15g\n", bb);   // prints 8.99999999999996

    return 0;
}


/*####################### EX 4 ##########################*/

/* toobig.c-exceeds maximum int size on our system */
#include <stdio.h>
int main(void)
{
    int i = 2147483647;
    unsigned int j = 4294967295;

    printf("%d %d %d\n", i, i+1, i+2);
    printf("%u %u %u\n", j, j+1, j+2);

    return 0;
}


/*####################### EX 5 ##########################*/

/* demonstrates the use of char data type (numerical type)
 * the assignment is done via numeric value
 * and the out is using literal char
 */

#include<stdio.h>

int main(){
    char c;  // declare char type variable
    c=100;   // store a value in the variable
    printf("Value is: %c\n",c); // display the value of the variable
    return 0;
}


/*####################### EX 6 ##########################*/

/* demontrates the use of real data type with double precision
 * for input and out
 */

#include <stdio.h>
int main(){
    double d;
    scanf("%lf", &d);  // read a double precision value (lf sppecifier)
    printf("%16.14lf\n", d); // display a double precision value specifying the number of characters for the output
    return 0;
}
```