```
#include <stdio.h>

int main(){

    char a = 'p';
    // Writing char a = p; will cause Mr C to search for
    // a variable named p
    // char a = "p" is also incorrect since a is not an array of char


        /*********** scanf and printf with characters ***********/
    // Use %c to operate scanf and printf with char
    scanf("%c", &a);
    printf("The character stored in a is %c\n", a);

        /*********** getchar and putchar ************/
        // getchar, putchar are shortcuts to read/print a single character
    char d;
    d = getchar();
    putchar(d); // Cannot print nice messages like in printf
        printf("\n");

        /*********** Typecasting with characters *************/
    // Typecasting (implicit vs explicit)
    printf("65 gets implicitly typecast %c\n", 65); // since ASCII code of A is 65 - implicit
typecasting
    printf("I can explicitly typecast to char to get 4th letter of alphabet %c\n", (char)(65 +
3)); // 4th capital letter of the alphabet

    // Implicit typecasting between float/char or float/int is dangerous in printf and scanf
    float f = 65.0;
    printf("65.0 implicit typecast to char %c\n", f); // This does not work
    printf("65.0 implicit typecast to int %d\n", f); // Even this does not work !!
    printf("65.0 explicit typecast to int %d\n", (int)f); // This works - do not rely on implicit
typecast if float/double involved with char/int
    printf("65.0 explicit typecast to char %c\n", (char)f); // This works - do not rely on
implicit typecast if float/double involved with char/int


        /*********** Exploiting the fact that char stored as int *************/
    // Can use cool tricks since char is stored as int
    char b = 'M';
    printf("%c in lower case is %c\n", b, b - 'A' + 'a');
    b = 'p';
    printf("%c in upper case is %c\n", b, b - 'a' + 'A');

    // Decimal digits when represented as characters vs as integers
    char c = '7';
    printf("I can get the value %d from the character %c\n", c - '0', c);

    // Tip: print char as int to debug
    d = 65;
    if(d = 32){
        printf("Character d is storing the alphabet %c\n", d);
        printf("Character d is storing the ASCII value %d\n", d); // Print the char as an int to
find out what went wrong
    }

    // CAUTION: Be careful when working with %c and %d, %f etc in scanf
    // When we only had %d, %f, %ld, %lf, all whitespaces got skipped. But %c will not skip any
whitespaces. It will read whitespace as a character

    int p, q;
    char e;
    scanf("%d%d%c", &a, &b, &c);
    printf("%d %d %c\n", a, b, c);
    // Give the input
    // 10 20 p
    // all separated by a space, and see what happens
    // To debug, print char as int to get the ASCII value of character
    printf("%d %d %d\n", a, b, c);

        // When used in arithmetic, relational, logical expressions,
        // ASCII (integer) value of character gets used
        int r = 'A' + 1; // 66
        if('A' < 'C') // ('A" < 'C') will be evaluated as (65 < 67)
                printf("A comes before C in the alphabet\n");

    return 0;
}
```