Lab 2: Characters, Strings, Operators, and Library Functions

Question 1

```
#include <iostream>
#include <string>

using namespace std;
int main (void){
string name;
cout << "Enter your full name: ";
getline(cin, name);
cout << "Welcome, " << name << endl;
return 0;
}
```

The cin only reads the first as when it encounters a white space it ignore everything after that, to fix that we need to use getline function which takes in the whitespaces as well.

Question 2

The output is acegd
because cin.ignore() ignores the very next element and keeps on going until we hit the character d giving us the output

Question 3

The output is 060566

Question 4

cout << static_cast(x << 3) << endl;

1. the first output would be 520 because the binary representation of 65 is 01000001 and when we shift towards left by 3, we add 3 zeroes on the right making the binary number 00101000 which is 520 in base 10.

cout << static_cast(x & 15) << endl;

2. the output would be 1 because binary representation of 65 is 01000001 and for 15 it is 00001111 and we use the & operator on it

01000001 & 00001111 it becomes 00000001 which is 1 in decimal

cout << static_cast(x | -50) << endl;

3. the output would be –1 because the binary representation of 65 is 01000001 and for –50 it is 11001110 and we use OR operator on it
01000001 | 11001110 it becomes using 2's complement –1

cout << static_cast(x >> 1) << endl;

4. The output would be 32 because the binary representation of 65 is 01000001 and when we shift towards right by 1, we add one 1 on the left side of the binary making it 00100000 which is 32 in base 10

cout << static_cast(x ^ -100) << endl;

5. the output would be 221 because binary representation of 65 is 01000001 and for - 100 it is 10011100 and we use the ^ operator on it
01000001 ^10011100 it becomes 11011101 which is 221 in base10

cout << static_cast(~x) << endl;

6. the output would be –66 as we would flip all the bits of 65 (01000001) making it 10111110 which is –66.

cout << static_cast((~x) + 1) << end

7. Since from above we already know that ~x is –66 adding 1 to it will make it –65. Hence, the output is –65

Question 5

a. True (or 1)
b. Nothing will happen since it is not a alphabet, it will remain same, hence "?"
c. "d"
d. C
e. Since it is an int, it will truncate the value after the decimal and output would be 1

f. The range would be from 0 to 12 but for range from –4 to –1, there would be very large number
g. “ABCD”
h. 7
i. 128 (2 ^ 7)
j. As it is a short int, it will overflow , and the output would be 0