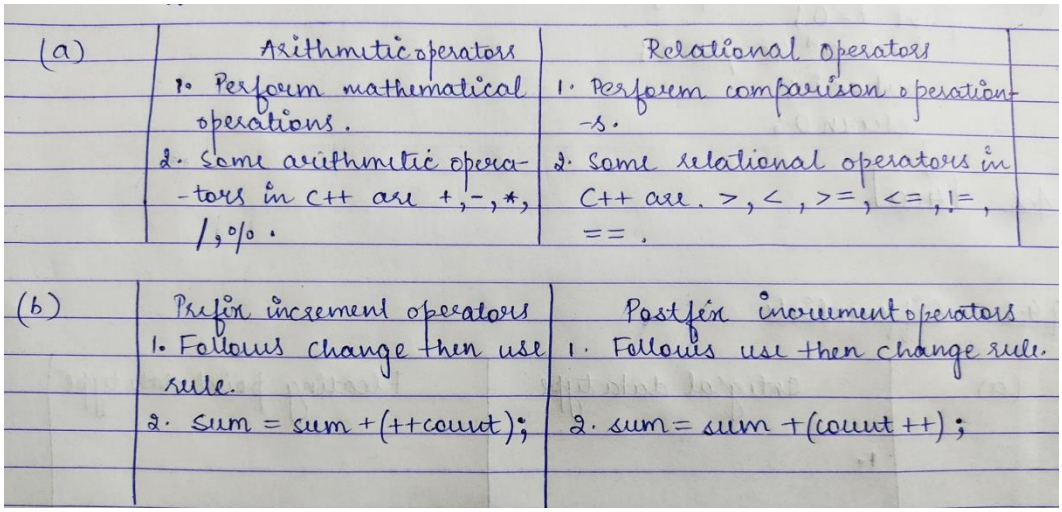
# CH3- ARITHMETIC & RELATIONAL OPERATORS- EXERCISE

***Submitted By-Sakshi***
***Roll No- 88001***

Q.3.1 Differentiate:



| (a) | Arithmetic operators | Relational Operators |
|---|---|---|
| | 1. Perform mathematical operations. | 1. Perform comparison operations. |
| | 2. Some arithmetic operators in C++ are +, -, *, /, %. | 2. Some relational operators in C++ are. >, <, >=, <=, !=, ==. |

| (b) | Prefix increment operators | Postfix increment operators |
|---|---|---|
| | 1. Follows change then use rule. | 1. Follows use then change rule. |
| | 2. sum = sum + (++count); | 2. sum = sum + (count++); |

Q.3.2 WAP to convert temperature from Celsius to Fahrenheit.

Program

```
/*temperature converter from celsius to fahrenheit*/
#include <bits/stdc++.h>
#include <math.h>
using namespace std;
int main()
{float tempCel,tempFahr;
cout<<"Enter value of temp.in celsius: ";
cin>>tempCel;
tempFahr=(9*tempCel)/5+32;
cout<<"temperature in fahrenheit is: "<<tempFahr;
return 0;

}
```

Output

```
Enter value of temp.in celsius: 37.22222
temperature in fahrenheit is: 99
Process returned 0 (0x0)   execution time : 4.192 s
Press any key to continue.
```

Q.3.3 WAP to calculate simple interest.

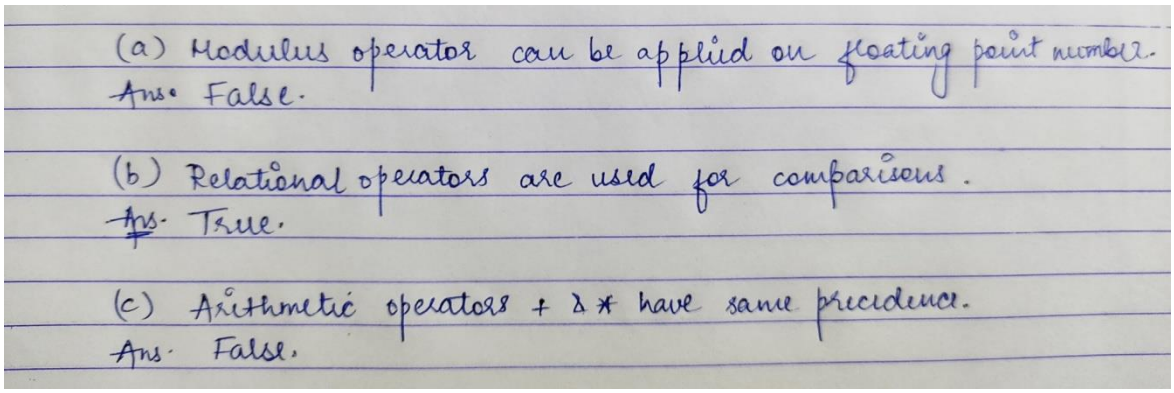
Program

```
#include <bits/stdc++.h>
#include <math.h>
using namespace std;
int main()
{float p,r,i;
int t;
cout<<"enter principal: ";
cin>>p;
cout<<"enter rate: ";
cin>>r;
cout<<"enter time";
cin>>t;
i=(p*r*t)/100;
cout<<"Principal interest is: "<<i;
return 0;
}
```

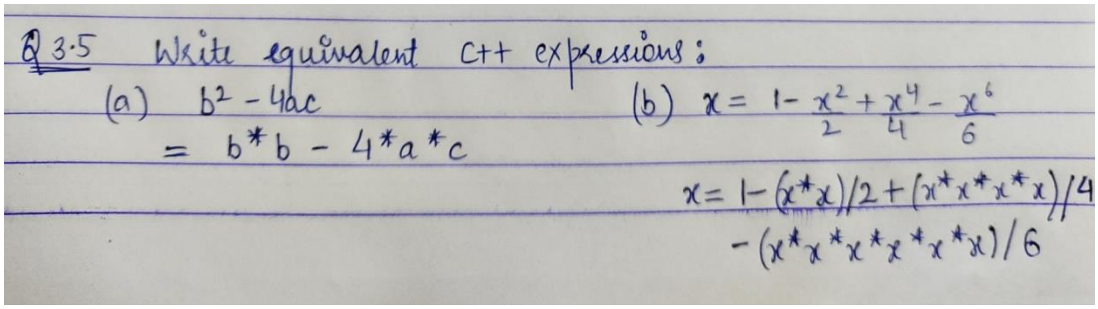
Output

```
enter principal: 100
enter rate: 20
enter time2
Principal interest is: 40
Process returned 0 (0x0)   execution time : 6.293 s
Press any key to continue.
```

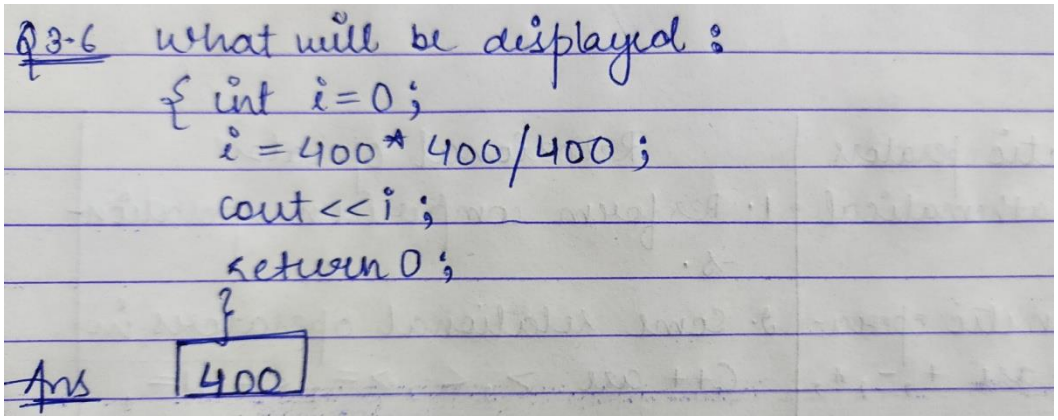
Q.3.4 Write true/false:



Q.3.5 Write equivalent expressions in c++:




Q.3.6 Display the output:




==+======+====+====+======+==

# CH 4- DATA TYPES EXERCISE

***Submitted By-Sakshi***
***Roll No- 88001***

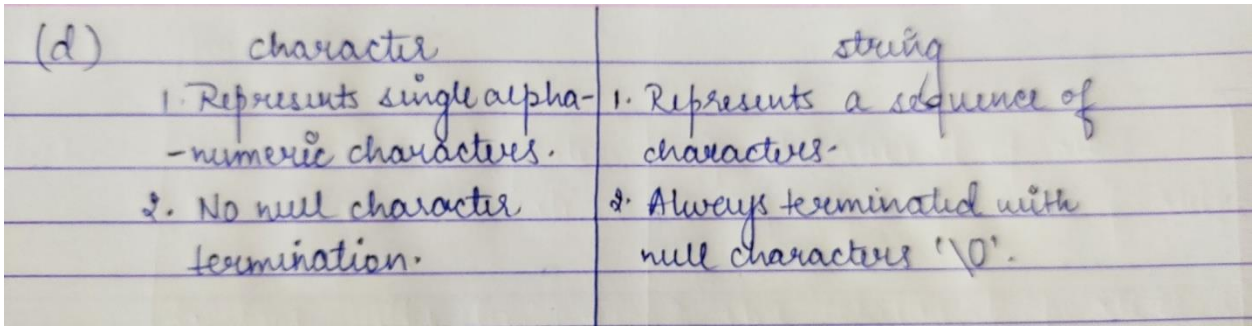
Q.4.1 Differentiate:

| Integral data type | Floating point data type |
|---|---|
| 1. It is a whole no. without a decimal point. | 1. It has a decimal point. |
| 2. Used for counting numbers. | 2. It is used for measuring quantities. |

(c)

| int | signed int | Unsigned int |
|---|---|---|
| 1. Represents a whole no. | 1. Represents signed value of any integral value. | 1. Represents unsigned value of integer. |
| 2. Range: -32768 to 32767 | 2. Range: -32768 to 32767 | 2. Range: 0 to 65555 |

| double | long double |
|---|---|
| 1. Represents double precision floating point value. | 1. Represents extended precision floating point value. |
| 2. Allocates 8 bytes to data. | 2. Allocates 12 bytes to data |
| 3. Range: 1.7e-308 to 1.7e+308 | 3. Range: 3.4e-4932 to 3.4e4932 |



| (d) character | string |
| --- | --- |
| 1. Represents single alpha-numeric characters. | 1. Represents a sequence of characters. |
| 2. No null character termination. | 2. Always terminated with null character '\0'. |

Q.4.2 Is it true that an unsigned int is twice as large as signed int? Why?

Ans Yes, it's true because unsigned int can have 65356 values and signed int can also have 65356 values, the only difference is that it stores negative values so its range extends from negative to positive integers.

Q.4.3 Write a short note on void data type.

Ans. Void type specifies an empty set of values. It is used as return type for functions. It declares explicitly a function. It has no values and no operations. Many programming languages need a type to define the lack of return value, so void is used.

Q.4.4 What is the use of modifiers in data types?

Ans. It alters the meaning of base data type to fit various situations more precisely. Modifiers include:

(a)signed (b)unsigned (c)long (d)short

Q.4.5 Give output:

```
{
cout<<7+7/7.0;
return (0);
}
```

Ans. 8

===+===+===+===+===+===+===+===+===+===

# CH 4- DATA TYPES EXERCISE

***Submitted By-Sakshi***
***Roll No- 88001***

Q.5.1 What are the entities called whose values can be changed?
Ans Variables

Q.5.2 State true/false:

(a)In an assignment expression, the type casting of variable on the left side can be done.
True
(b)There is never a loss of information when an integer variable is divided by another integer.
False
(c)By type casting an expression, the accuracy of the result can be increased.
True

Q.5.3Give output:

```
{
float f;
f=5/2*float(7)/(int)3.5;
cout<<f;
return 0;
}
```

Ans. 4.667

Q.5.4 Sometimes when an integer value is moved to a character variable and then moved back to integer variable, we get negative value. Why?

Ans. Yes, if the integral value of character is out of range then it may be possible that we may end up getting a negative value.

Q.5.5 Display the output:

```
{
int i=2,j=5;
float f1,f2;
f1=j/i;
f2=float(j)/i;
cout<<f1<<"\n"<<f2;
return 0;
}
```

Ans. 2, 2.5