2) A/ In C++, for loops, while loops and do-while loops are used for repetition. Each of these loops has its own syntax and cases. The differences are given below with examples:

For Loop:

1. The loop initialization, condition and update are all defined within the loop's header.
2. The loop condition checked before entering the loop.

Example:

```
for(int i=1; i<=5; i++){
    cout << i << " ";
}
```

output: 1 2 3 4 5

While Loop:

1. This loop is used when developer/programmer wants to execute a block of code as long as a certain condition is true.
2. The condition is checked before entering the loop.

Example:

```
int count = 1;
while (count <=5) {
    cout << count << " ";
    count++;
}
```

output: 1 2 3 4 5

Do-while Loop:

1. This loop is used when developer/programmer wants to execute a block of code at least once and then continue to execute it as long as a certain condition is true.
2. The condition is checked after the loop body, ensuring that the loop body always executes at least onee.

Example:

```
int i=1;
do {
    cout << i << " ";
    i++;
}
while (i<=5);
```

output: 1 2 3 4 5

—o—

B/ The 'break' and 'continue' keywords are used to control the flow of loops.

1. Break:

The 'break' keyword is used to exit a loop prematurely when a certain condition is met.

```
for(int i=1 ; i<=10 ; i++) {
    if(i==5) {
        break;
    }
    cout << i << " ";
}
```

Output: 1 2 3 4

2. Continue:

The 'continue' keyword is used to skip the current flow of a loop and move to the next flow without executing the remaining code in the current flow.

```
for(int i=1 ; i<=5; i++) {
    if(i==2) {
        continue;
    }
    cout << i << " ";
}
```

Output:

1 3 4 5

—o—

c) Nested if-else:

In programming code, if it have one if-else statement inside another if-else statement, creating a nested if-else structure.

It is useful when a programmer needed to check multiple conditions sequentially. Each inner if-else statement is executed based on the condition of the outer if-else statement.

example:

```
int num = 15
if(num > 10) {
    cout << "Number is greater than 10."
    << endl;
    if(num > 20) {
        cout << "Number is also greater
        than 20." << endl;
    }
    else {
        cout << "Number is not greater than
        20." << endl;
    }
}
else {
    cout << "Number is not greater than 10."
    << endl;
```

Nested Loop:

Nested loops are loops inside other loops. They are used for making multiple repeation of certain blocks of code or repeating two different codes / tasks.

Example:

```
for(int i=0 ; i<3 ; i++) {
    for(int j=0 ; j<i+1 ; j++) {
        cout << "*";
    }
    cout << endl;
}
```

Output:

```
*
**
***
```

Here inner loop prints "*" and outer loop prints the endline.

Q) Type casting refers to the process of converting the data type of an expression or a variable into different data type. Type casting are two types:

1. Explicit, 2. Implicit.

1. ~~Explicit~~ Implicit Type casting.

This is done automatically by the compiler when programmer mixes different data types in an expression or ~~va~~ variable. It converts the data type of one operand to match the data type of the other operand.

Example:

```
int a = 5;
float b = 2.5;
float result = a + b;
cout << result;
```

output: 7.5

Here, implicitly casts a to float data type for the addition.

2. Explicit Type casting :

This is performed explicitly by the programmer by using casting operators such like (int), (float), (double) etc. to change the data type of value.

Example:

```
float pi = 3.1416;
cout << (int)pi;
```

output: 3

Here, explicitly converts float data type to int data type.