SECJ 1013 PROGRAMMING TECHNIQUE 1 LAB
EXERCISE 2

Name: LOAI MOHAMMED MOHAMMED AL-SABAHI

Matric Number: A21MJ4003

(1) Based on the given declaration, evaluate the following expression either it is **TRUE** or **FALSE**.

```
bool found = true;
bool flag = false;
char ch = 'R';
double x_double = 22.0;
double y_double =200.0;
double num = 15.3;
int x_int = 22;
int y_int = 200;
```

a) !flag   flag is false so !flag = (true)

b) (num >= y_double) && (num >= x_double)   we have false statement AND false statement so (false)

c) (found || flag)   we have true OR false so the answer must be (true)

d) (x_int == y_int)   x_int is not equal to y_int   so the answer is (false)

e) 'N' < = ch && ch <= 'Z'

The ASCII value of 'N' is 74

The ASCII value of 'R' is 82

The ASCII value of 'Z' is 90

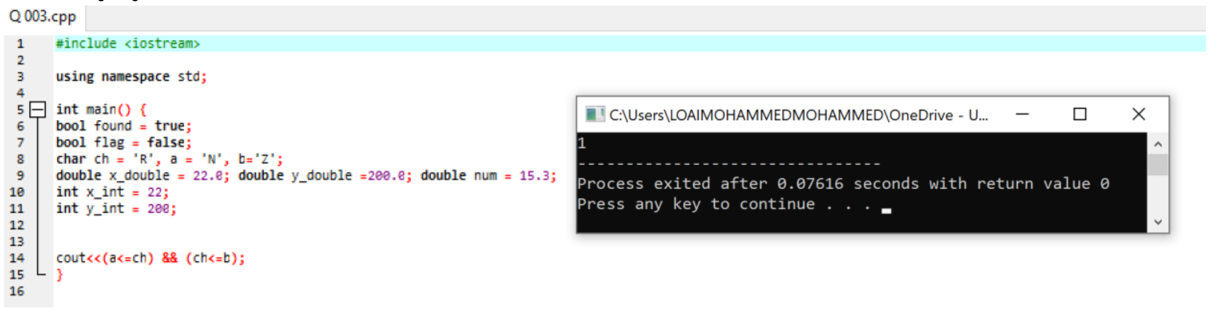
So we get

74 < = 82 true

82 < = 90 true

So true AND true is (true)

Note, this last output will not be executed unless you put brackets as illustrated in the picture below👇👇



Q 003.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
bool found = true;
bool flag = false;
char ch = 'R', a = 'N', b='Z';
double x_double = 22.0; double y_double =200.0; double num = 15.3;
int x_int = 22;
int y_int = 200;


cout<<(a<=ch) && (ch<=b);
}
```

C:\Users\LOAIMOHAMMEDMOHAMMED\OneDrive - U...

```
1
--------------------------------
Process exited after 0.07616 seconds with return value 0
Press any key to continue . . .
```

(2) What is the output of the following intermediate C++ codes?

a)

```cpp
int n=8;

if (!n>9)
cout << "Yes";
else
cout << "No";
```

The output is “No”
Because !n is equal to zero

b)

```cpp
int main()
{
      for(int i=3; i>=1; i--){
            string output(i, '+');
            cout << output << "\n";
      }

      for (int i=1; i<=3; i++){
            string output(i, '+');
            cout << output << "\n";
      }
return 0;
}
```

First loop:

First iteration i is equal to 3 so it will print +++

second iteration i is equal to 2 so it will print ++

Third iteration i is equal to 1 so it will print +

And then stops looping because the condition is no longer true

Second Loop:

First iteration i is equal to 1 so it will print +

Second iteration i is equal to 2 so it will print ++

Third iteration i is equal to 3 so it will print +++

And then stops because looping the condition is no longer true

To conclude the output will be:

```
+++

++

+

+

++

+++
```

c)

```
int main() {
int x1 = 2, x2 = 5, m = 13;
bool b1, b2, b3=true;
b1 = x1 == x2; // false
b2 = x1 < x2; // true
cout << "b1 = " << b1 << " and b2 =
" << b2 << "\n";
if (b3 == true) cout << "Yes" <<
"\n";
else cout << "No" << "\n";
int x3 = false + 3 * m - b3;
cout << x3;
return 0;}
```

First b1 will be equal to 0 because false is represented as 0 in binary numbers

and b2 will be equal to 1 because true is represented as 1 in binary numbers

Second b3 == true will print out Yes because the condition is true

Third we have type conversion from Boolean to integer and we must also take into consideration the operator precedence so

```
x3 = false + 3 * m - b3
will be
x3 = 0 + (3 * 13) - 1
so
```
`x3 = 38`

d) `for ( int i = 10, j = 13 ; i * j >= 130 ; i++, j-- ) { cout << "i * j = " << (i * j) <<"\t"; }`

In first iteration i will be 10 and j will be 13, so 10 * 13=130 which makes the condition

`i * j >= 130` true so it will be executed

In second iteration i will be 11 and j will be 12, so 11 * 12=132 which makes the condition

`i * j >= 130` true so it will be executed

In third iteration i will be 12 and j will be 11, so 12 * 11=132 which makes the condition `i * j >= 130` true so it will be executed

In forth iteration i will be 13 and j will be 10, so 13* 10=130 which makes the condition

`i * j >= 130` true so it will be executed

BUT In fifth iteration i will be 14 and j will be 9, so 14 * 9=126 which makes the condition

`i * j >= 130` FALSE so it will NOT be executed and it will stop looping

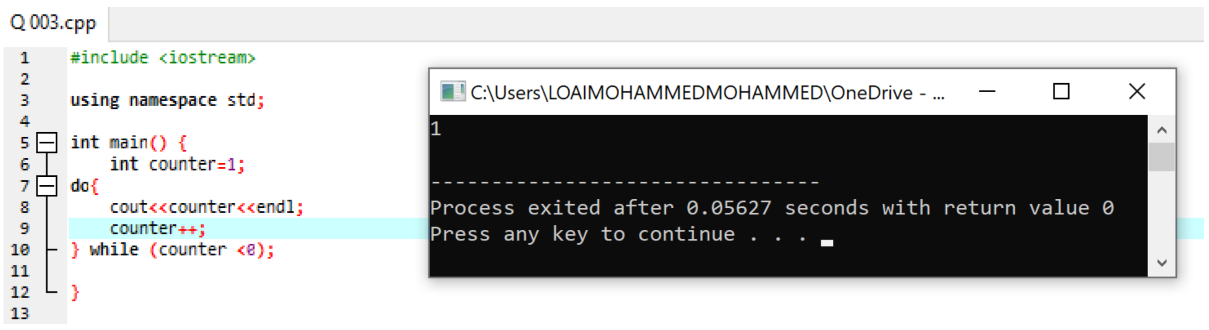
to conclude the output will be:

```
i * j = 130

i * j = 132

i * j = 132

i * j = 130
```

(6) Elaborate the difference between while, for and do … while statement?

(do…while) loop will execute the loop at least one time before checking the condition, that's why it is called posttest loop

For example, let's analyze this code



Q 003.cpp

```
#include <iostream>

using namespace std;

int main() {
    int counter=1;
do{
    cout<<counter<<endl;
    counter++;
} while (counter <0);

}
```

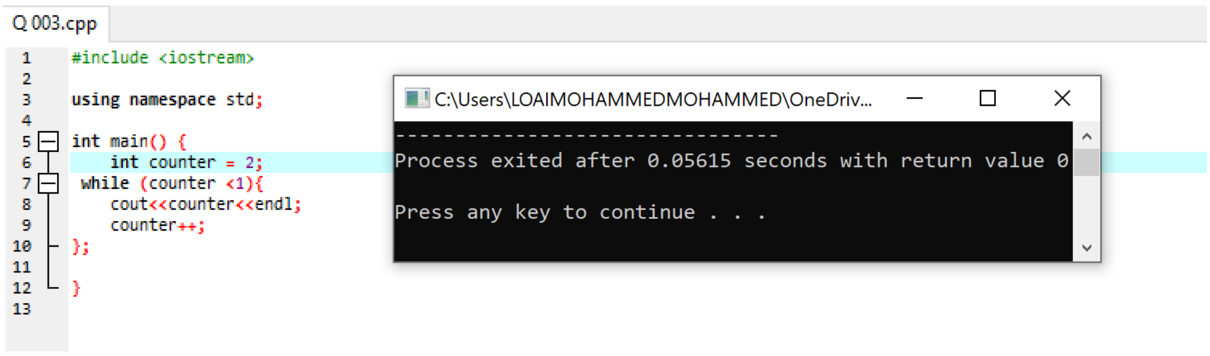
C:\Users\LOAIMOHAMMEDMOHAMMED\OneDrive - ...

```
1

--------------------------------
Process exited after 0.05627 seconds with return value 0
Press any key to continue . . .
```

We notice that (counter = 1) was executed even though that the condition (counter < 0) is false.

FOR and WHILE loops are called pretest and they will check the condition before executing the code.

For example, let's analyze these codes

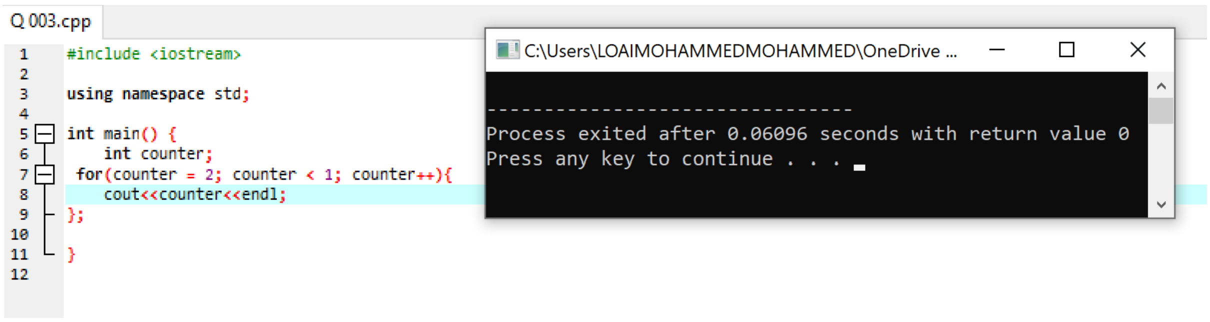


Q 003.cpp

```
#include <iostream>

using namespace std;

int main() {
     int counter = 2;
 while (counter <1){
     cout<<counter<<endl;
     counter++;
};

}
```

C:\Users\LOAIMOHAMMEDMOHAMMED\OneDriv...

```
--------------------------------
Process exited after 0.05615 seconds with return value 0

Press any key to continue . . .
```



Q 003.cpp

```
#include <iostream>

using namespace std;

int main() {
     int counter;
 for(counter = 2; counter < 1; counter++){
     cout<<counter<<endl;
};

}
```

C:\Users\LOAIMOHAMMEDMOHAMMED\OneDrive ...

```

--------------------------------
Process exited after 0.06096 seconds with return value 0
Press any key to continue . . .
```

We notice that there is no output, and that’s because FOR and WHILE loops will check the condition before they execute the code. So here the condition (counter < 1) is false so there is no code to be executed.

The small difference between FOR and WHILE loops is that when we use FOR loop, we already know how many times we want the code to be executed.