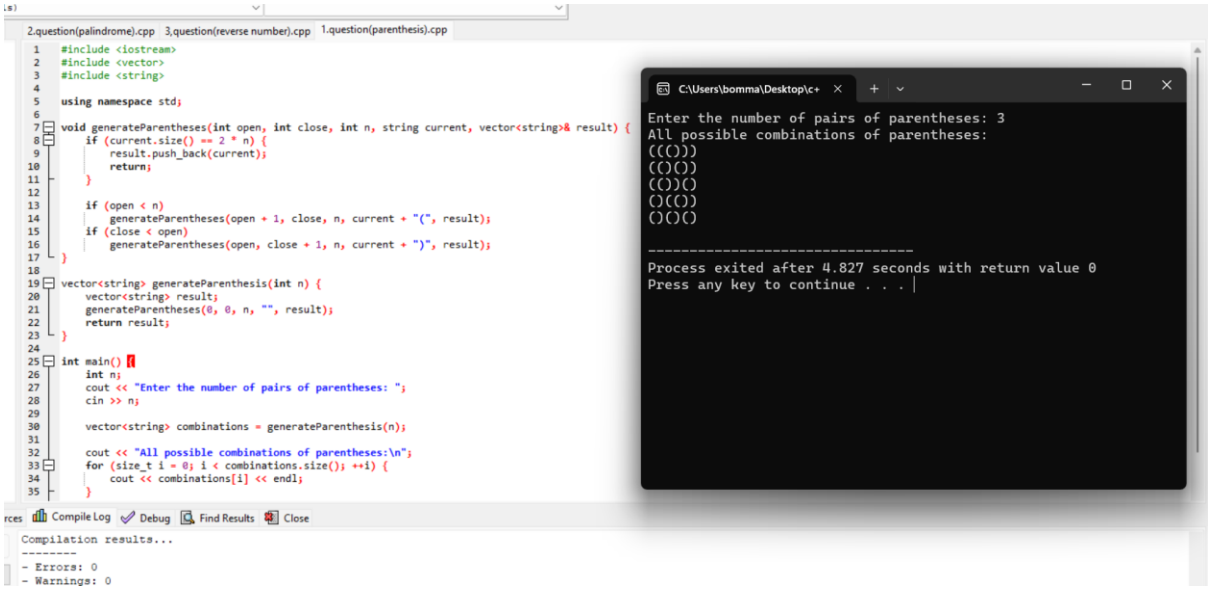
MODEI EXAM OUTPUTS:

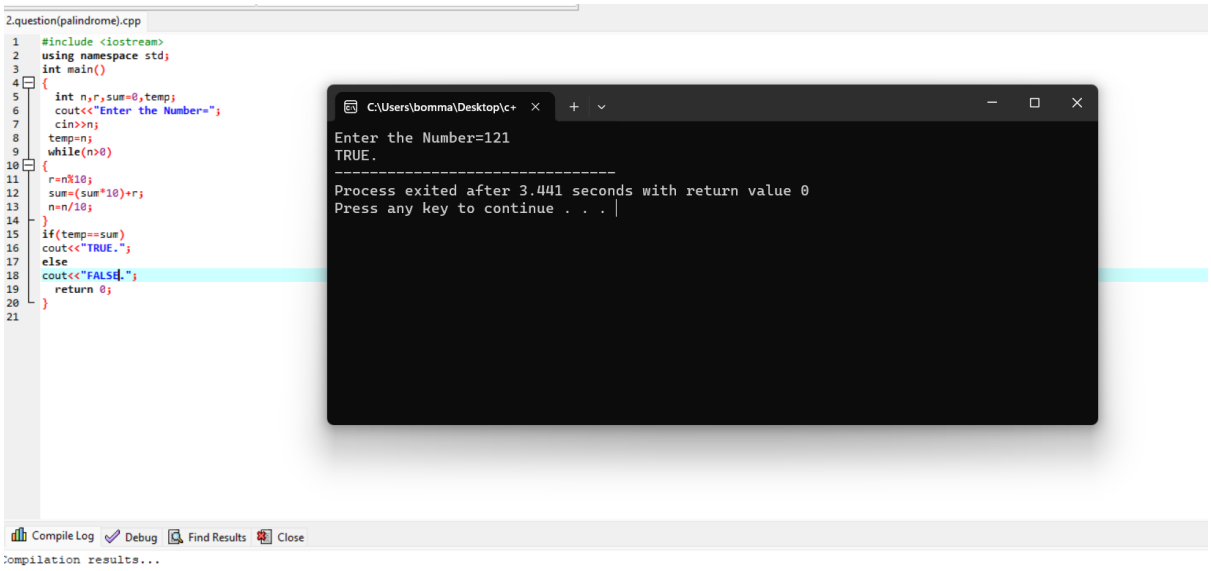
1.paranthesis



```
2.question(palindrome).cpp  3.question(reverse number).cpp  1.question(parenthesis).cpp

#include <iostream>
#include <vector>
#include <string>

using namespace std;

void generateParentheses(int open, int close, int n, string current, vector<string>& result) {
    if (current.size() == 2 * n) {
        result.push_back(current);
        return;
    }

    if (open < n)
        generateParentheses(open + 1, close, n, current + "(", result);
    if (close < open)
        generateParentheses(open, close + 1, n, current + ")", result);
}

vector<string> generateParenthesis(int n) {
    vector<string> result;
    generateParentheses(0, 0, n, "", result);
    return result;
}

int main() {
    int n;
    cout << "Enter the number of pairs of parentheses: ";
    cin >> n;

    vector<string> combinations = generateParenthesis(n);

    cout << "All possible combinations of parentheses:\n";
    for (size_t i = 0; i < combinations.size(); ++i) {
        cout << combinations[i] << endl;
    }
}

Compile Log  Debug  Find Results  Close
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

```
C:\Users\bomma\Desktop\c+
Enter the number of pairs of parentheses: 3
All possible combinations of parentheses:
((()))
(()())
(())()
()(())
()()()

--------------------------------
Process exited after 4.827 seconds with return value 0
Press any key to continue . . .
```

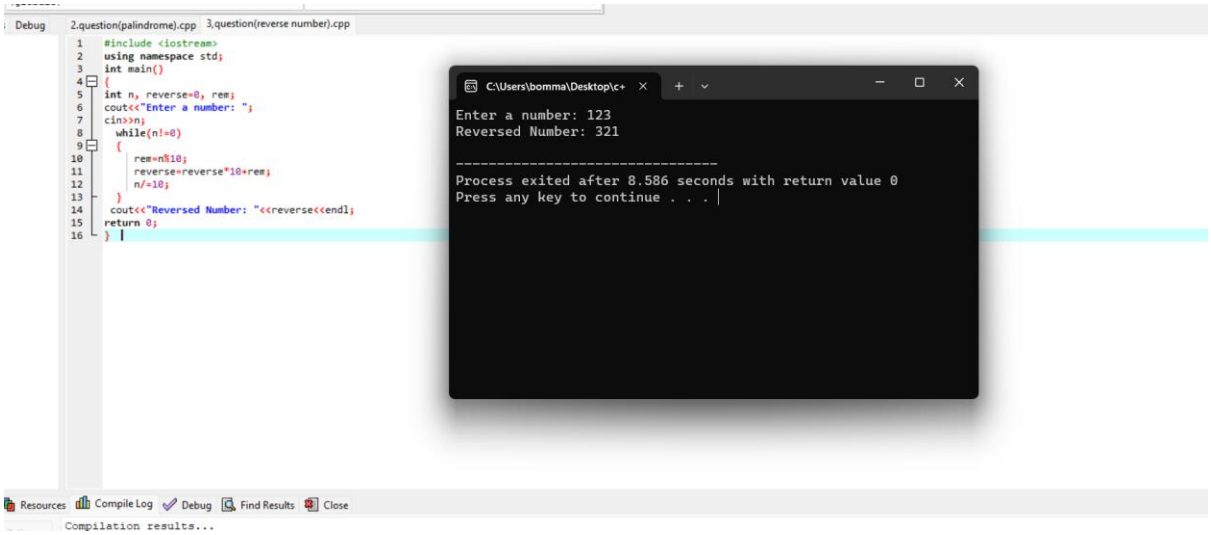
2.palindrome



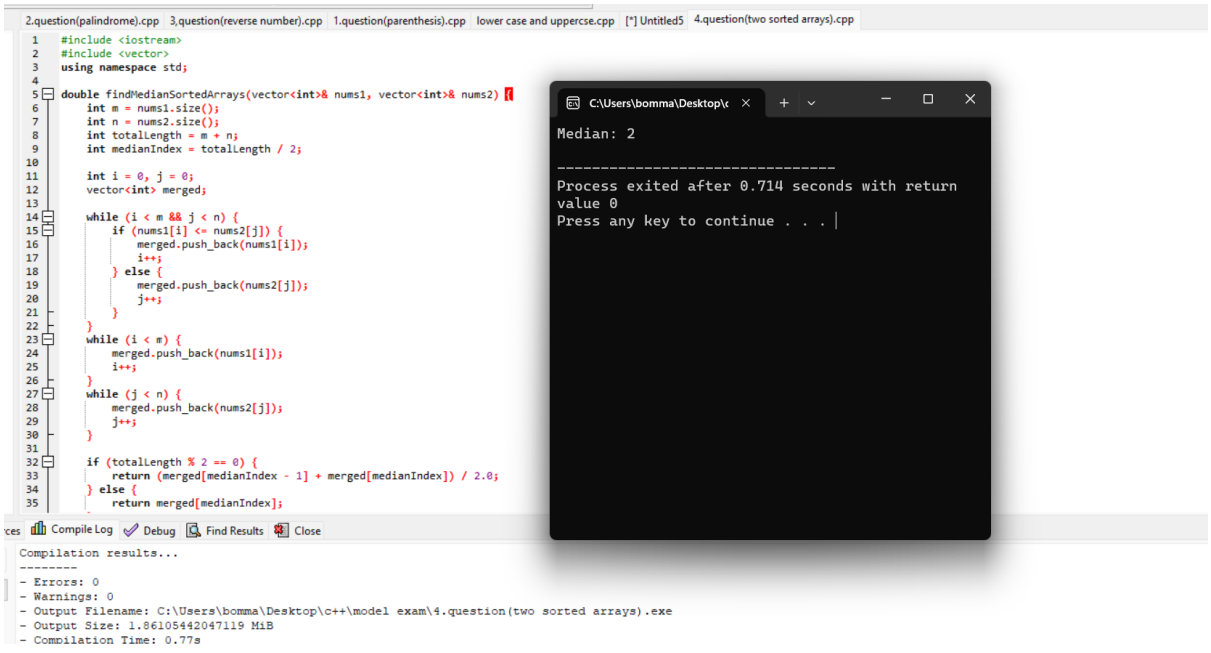
```
2.question(palindrome).cpp

#include <iostream>
using namespace std;
int main()
{
  int n,r,sum=0,temp;
  cout<<"Enter the Number=";
  cin>>n;
 temp=n;
 while(n>0)
{
 r=n%10;
 sum=(sum*10)+r;
 n=n/10;
}
if(temp==sum)
cout<<"TRUE.";
else
cout<<"FALSE.";
  return 0;
}

Compile Log  Debug  Find Results  Close
Compilation results...
```

```
C:\Users\bomma\Desktop\c+
Enter the Number=121
TRUE.
--------------------------------
Process exited after 3.441 seconds with return value 0
Press any key to continue . . .
```

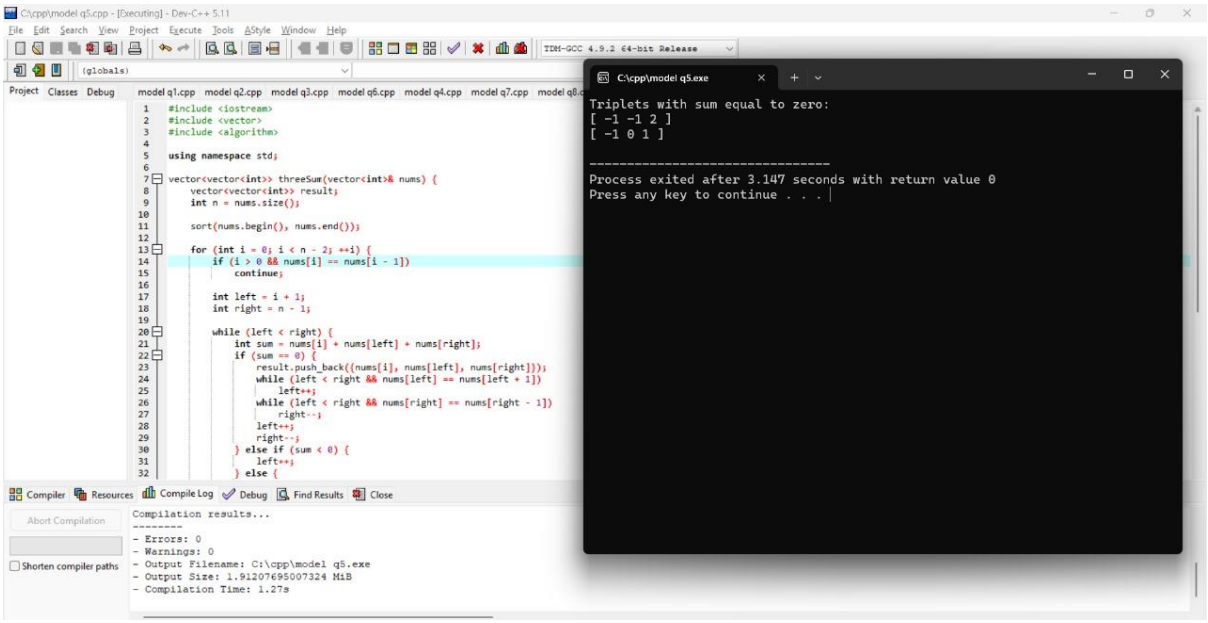
3.Reverse number



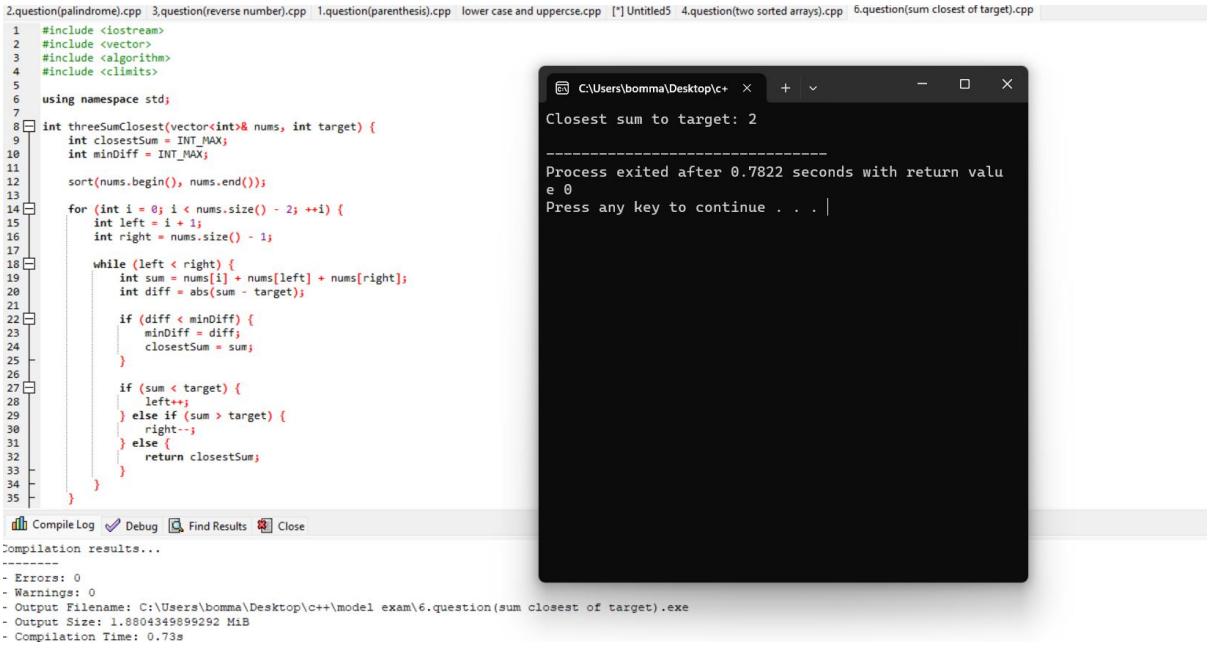
4.Two sorted arrays of num1 and num2:



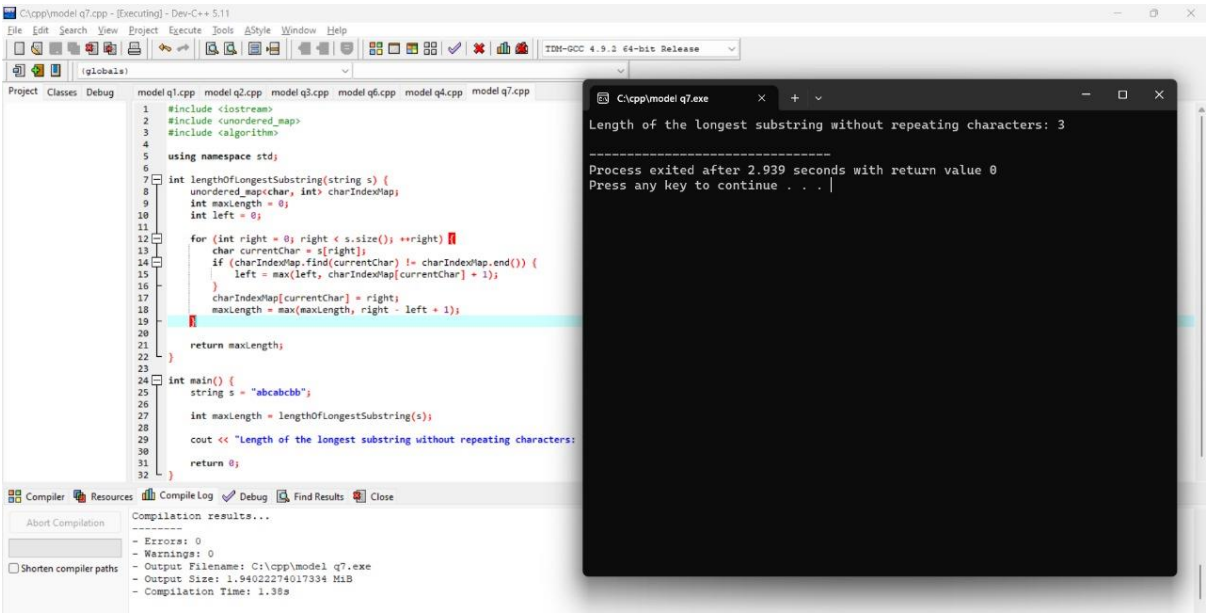
5.triplets:



6.sum is closest to target:



7.length of longestsubstring:



8.string s and apattern p, implement regular expression: