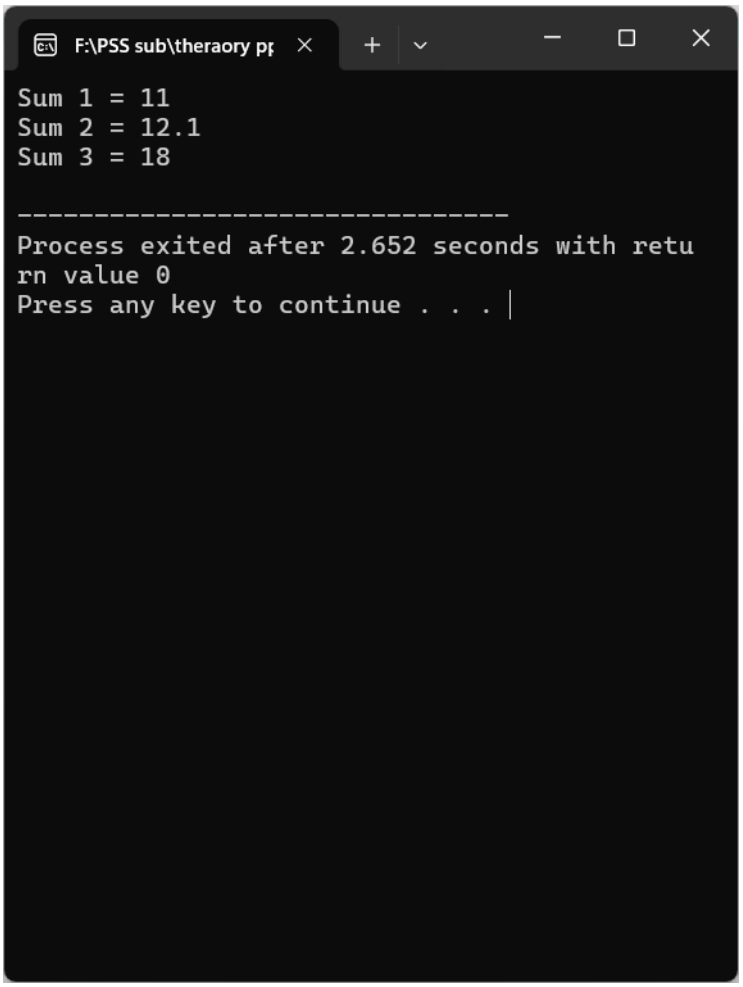
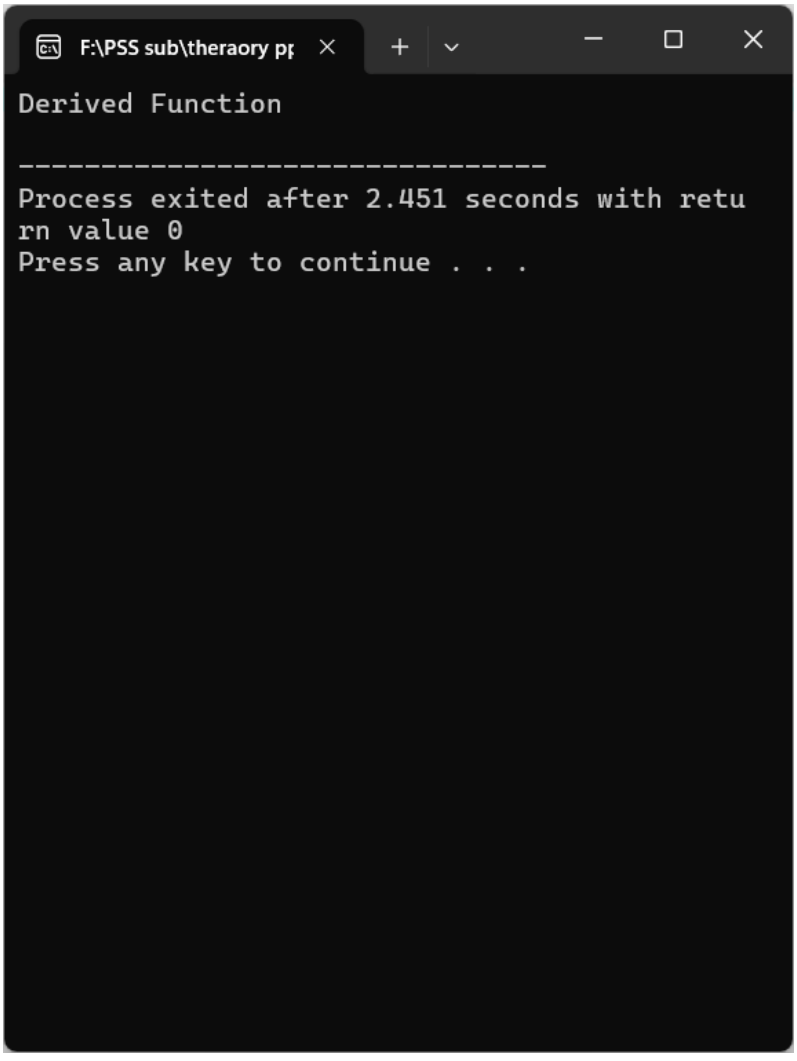
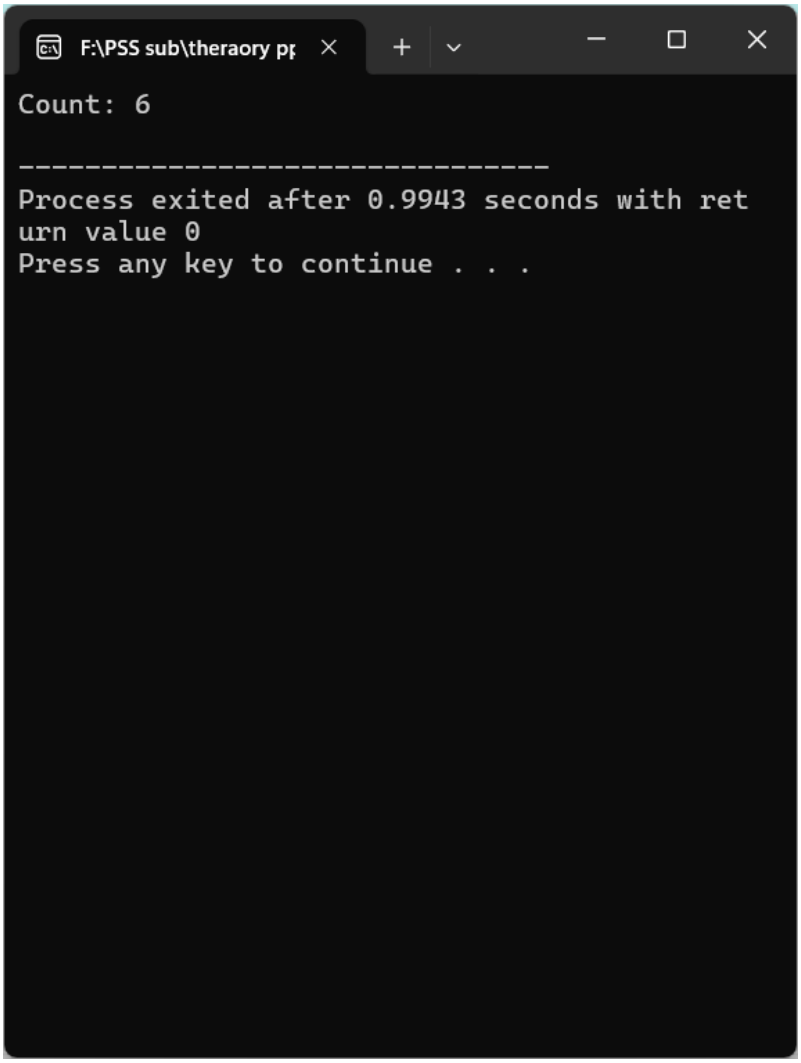
```cpp
//function overloading programm
#include <iostream>
using namespace std;
// Function with 2 int parameters
int sum(int num1, int num2) {
return num1 + num2;
}
// Function with 2 double parameters
double sum(double num1, double num2) {
return num1 + num2;
}
// Function with 3 int parameters
int sum(int num1, int num2, int num3) {
return num1 + num2 + num3;
}
int main() {
// Call function with 2 int parameters
cout << "Sum 1 = " << sum(5, 6) << endl;
// Call function with 2 double parameters
cout << "Sum 2 = " << sum(5.5, 6.6) << endl;
// Call function with 3 int parameters
cout << "Sum 3 = " << sum(5, 6, 7) << endl;
return 0;
}
```



```cpp
//function overriding program
#include <iostream>
using namespace std;
class Base {
public:
virtual void print() {
cout << "Base Function" << endl;
}
};
class Derived : public Base {
public:
void print() {
cout << "Derived Function" << endl;
}
};
int main() {
Derived derived1;
// Call print() function of Derived class
derived1.print();
return 0;
}
```



```cpp
//operator overloading program
#include <iostream>
using namespace std;
class Count {
private:
int value;
public:
// Constructor to initialize count to 5
Count() : value(5) {}
// Overload ++ when used as prefix
void operator ++() {
value = value + 1;
}
void display() {
cout << "Count: " << value << endl;
}
};
int main() {
Count count1;
// Call the "void operator ++()" function
++count1;
count1.display();
return 0;
}
```



```cpp
//virtual function program
#include <iostream>
using namespace std;
class Base {
public:
virtual void print() {
cout << "Base Function" << endl;
}
};
class Derived : public Base {
public:
void print() {
cout << "Derived Function" << endl;
}
};
int main() {
Derived derived1;
// pointer of Base type that points to derived1
Base* base1 = &derived1;
// calls member function of Derived class
base1->print();
return 0;
}
```