7. **Define a friend function addTime() with objects as arguments and return the sum of two objects. Show the values of each object and their sum as output.**

PROGRAM

```
#include<iostream>
using namespace std;

class Time {
   private:
         int hour, minute;
   public:
         Time() {
            hour = 0;
            minute = 0;
         }
         Time(int hour, int minute) {
            this->hour = hour;
            this->minute = minute;
         }
         friend Time addTime(Time t1, Time t2);
         void display() {
             cout<<"Hour: "<<hour<<endl<<"Minute: "<<minute<<endl;
         }
};

Time addTime(Time t1, Time t2) {
   Time temp;
   temp.hour = t1.hour + t2.hour;
   temp.minute = t1.minute + t2.minute;
   if (temp.minute >= 60) {
                  temp.hour += temp.minute / 60;
                  temp.minute %= 60;
   }
   return temp;
}

int main() {
   Time t1(10, 50), t2(1, 11), t3;
   t3 = addTime(t1, t2);
   cout<<"Time 1: "<<endl;
   t1.display();
   cout<<"Time 2: "<<endl;
   t2.display();
   cout<<"Time 3: "<<endl;
   t3.display();
   return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\friend_function.exe
Time 1:
Hour: 10
Minute: 50
Time 2:
Hour: 1
Minute: 11
Time 3:
Hour: 12
Minute: 1
```

8. Write different programs to implement passing by reference and passing by value in C++.

PROGRAM

```
#include <iostream>
using namespace std;
void passByValue(int a) {
    a = a * 2;
    cout << "Inside passByValue function: " << a << endl;
}
void passByReference(int &a) {
    a = a * 2;
    cout << "Inside passByReference function: " << a << endl;
}
int main() {
    int x = 5;
    cout << "Before passByValue function: " << x << endl;
    passByValue(x);
    cout << "After passByValue function: " << x << endl;
    cout << "Before passByReference function: " << x << endl;
    passByReference(x);
    cout << "After passByReference function: " << x << endl;
    return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\pass by value reference.exe
Before passByValue function: 5
Inside passByValue function: 10
After passByValue function: 5
Before passByReference function: 5
Inside passByReference function: 10
After passByReference function: 10
```

9. Write different programs to implement different storage classes (auto, register, extern and static) in C++ with its output.

PROGRAM

```
#include <iostream>
using namespace std;

int externVar = 100;

void displayExtern() {
    cout << "Inside displayExtern function: externVar = " << externVar << endl;
}

void demonstrateAuto() {
    auto autoVar = 10;
    cout << "Inside demonstrateAuto function: autoVar = " << autoVar << endl;
}
void demonstrateRegister() {
    register int registerVar = 20;
    cout << "Inside demonstrateRegister function: registerVar = " << registerVar << endl;
}
void demonstrateStatic() {
    static int staticVar = 30;
    cout << "Inside demonstrateStatic function: staticVar = " << staticVar << endl;
    staticVar++;
}
```

```
int main() {
    cout << "Inside main function: externVar = " << externVar << endl;
    demonstrateAuto();
    demonstrateRegister();
    demonstrateStatic();
    demonstrateStatic();
    externVar = 200;
    displayExtern();
    return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\static extern keywords.exe
Inside main function: externVar = 100
Inside demonstrateAuto function: autoVar = 10
Inside demonstrateRegister function: registerVar = 20
Inside demonstrateStatic function: staticVar = 30
Inside demonstrateStatic function: staticVar = 31
Inside displayExtern function: externVar = 200
```

10. Write a C++ program to illustrate dynamic allocation and de-allocation of memory using new and delete.

PROGRAM

```
#include <iostream>
using namespace std;
int main() {
    int* ptr = new int;
    *ptr = 42;
    cout << "Value of the single integer: " << *ptr << endl;
    delete ptr;
    int size = 5;
    int* arr = new int[size];
    for (int i = 0; i < size; ++i) {
        arr[i] = i * 10;
    }
```

```
    cout << "Values in the array: ";
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
    delete[] arr;
    return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\dma.exe
Value of the single integer: 42
Values in the array: 0 10 20 30 40
```

11. Write a program using dynamic memory allocation to get input an array of numbers and find the sum of N numbers stored in the array using a function to compute the sum.

PROGRAM

```
#include <iostream>
using namespace std;
int main() {
    int* ptr = new int;
    *ptr = 42;
    cout << "Value of the single integer: " << *ptr << endl;
    delete ptr;

    int size = 5;
    int* arr = new int[size];
    for (int i = 0; i < size; ++i) {
        arr[i] = i * 10;
    }
```

```
    cout << "Values in the array: ";
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
    delete[] arr;
    return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\dma.exe
Value of the single integer: 42
Values in the array: 0 10 20 30 40
```

12. Write a program to implement user defined constructor and copy constructor.

PROGRAM

```
#include <iostream>
using namespace std;
class MyClass {
private:
    int value;
public:
    MyClass(int v) {
            value = v;
        cout << "User-defined constructor called with value: " << value << endl;
    }

    MyClass(const MyClass &other) {
            value = other.value;
        cout << "Copy constructor called, copied value: " << value << endl;
    }
```

```
    void display() {
        cout << "Value: " << value << endl;
    }
};

int main() {
    MyClass obj1(10);
    obj1.display();

    MyClass obj2 = obj1;
    obj2.display();

    MyClass obj3(obj1);
    obj3.display();

    return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\userdefined & copy constructor.exe
User-defined constructor called with value: 10
Value: 10
Copy constructor called, copied value: 10
Value: 10
Copy constructor called, copied value: 10
Value: 10
```

13. Write a program to illustrate constructor overloading in C++.

PROGRAM

```
#include <iostream>
#include <string>
using namespace std;
class MyClass {
private:
    int a;
    double b;
    string c;
public:
    MyClass() {
        a = 0;
        b = 0.0;
        c = "default";
        cout << "Default constructor called" << endl;
    }
    MyClass(int x) {
        a = x;
        b = 0.0;
        c = "default";
        cout << "Constructor with one parameter called, a = " << a << endl;
    }

    MyClass(int x, double y) {
        a = x;
        b = y;
        c = "default";
        cout << "Constructor with two parameters called, a = " << a << ", b = " << b << endl;
    }
```

```
    void display() const {
        cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
    }
};
int main() {
    MyClass obj1;
    obj1.display();
    MyClass obj2(10);
    obj2.display();
    MyClass obj3(20, 30.5);
    obj3.display();
    return 0;
}
```

OUTPUT

```
C:\Users\suresh\OOP\userdefined & copy constructor.exe
Default constructor called
a = 0, b = 0, c = default
Constructor with one parameter called, a = 10
a = 10, b = 0, c = default
Constructor with two parameters called, a = 20, b = 30.5
a = 20, b = 30.5, c = default

--------------------------------
```