|  |
| --- |
| Question 4 |

Predict the output of following C++ program.

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;    class A  {  private:      int x;  public:      A(int \_x)  {  x = \_x; }      int get()  { return x; }  };    class B  {      static A a;  public:     static int get()     {  return a.get(); }  };    int main(void)  {      B b;      cout << b.get();      return 0;  } | |
| A | 0 | |
| B | Linker Error: Undefined reference B::a | |
| C | Linker Error: Cannot access static a | |
| D | Linker Error: multiple functions with same name get() | |

Explanation:

There is a compiler error because static member a is not defined in B. To fix the error, we need to explicitly define a. The following program works fine.

#include <iostream >

using namespace std;

class A

{

private:

int x;

public:

A(int \_x) { x = \_x; }

int get() { return x; }

};

class B

{

static A a; // just a declaration and not defined yet.

public:

static int get()

{ return a.get(); }

};

A B::a(0);

int main(void)

{

B b;

cout << b.get();

return 0;

}

|  |
| --- |
| Question 5 |

|  |  |
| --- | --- |
| #include<iostream>  using namespace std;    class Test  {  private:      static int count;  public:      Test& fun();  };    int Test::count = 0;    Test& Test::fun()  {      Test::count++;      cout << Test::count << " ";      return \*this;  }    int main()  {      Test t;      t.fun().fun().fun().fun();      return 0;  } | |
| A | Compiler Error |
| B | 4 4 4 4 |
| C | 1 1 1 1 |
| D | 1 2 3 4 |

Explanation:

Static members are accessible in non-static functions, so no problem with accessing count in fun(). Also, note that fun() returns the same object by reference.

|  |
| --- |
| Question 6 |

Output of following C++ program?

|  |
| --- |
| #include <iostream>  class Test  {  public:      void fun();  };  static void Test::fun()  {      std::cout<<"fun() is staticn";  }  int main()  {      Test::fun();      return 0;  } |

|  |  |
| --- | --- |
| A | fun() is static |
| B | Empty Screen |
| C | Compiler Error |

Explanation:

The above program fails in compilation and shows below error messages. [Error] cannot declare member function 'void Test::fun()' to have static linkage [-fpermissive] In function 'int main()': [Error] cannot call member function 'void Test::fun()' without object If the static function is to be defined outside the class then static keyword must be present in function declaration only not in the definition outside the class.

Following program is now correct.

#include <iostream>

class Test

{

public:

    static void fun();

};

void Test::fun()

{

    std::cout<<"fun() is staticn";

}

int main()

{

    Test::fun();

    return 0;

}

**Static Keyword in C++**

**Static Variables :** Variables in a function, Variables in a class  
**Static Members of Class :** Class objects and Functions in a class

**Static Variables**

**Static variables in a Function**: When a variable is declared as static, space for **it gets allocated for the lifetime of the program**. Even if the function is called multiple times, space for the static variable is allocated only once and the value of variable in the previous call gets carried through the next function call. This is useful for implementing [coroutines in C/C++](https://www.geeksforgeeks.org/coroutines-in-c-cpp/) or any other application where previous state of function needs to be stored.

**Static variables in a class**: As the variables declared as static are initialized only once as they are allocated space in separate static storage so, the static variables **in a class are shared by the objects.** There cannot be multiple copies of same static variables for different objects. Also because of this reason static variables cannot be initialized using constructors.

**Static Members of Class**

**Class objects as static**: Just like variables, objects also when declared as static have a scope till the lifetime of program.  
Consider the below program where the object is non-static.

|  |
| --- |
| // CPP program to illustrate  // when not using static keyword  #include<iostream>  using namespace std;    class GfG  {      int i;      public:          GfG()          {              i = 0;              cout << "Inside Constructor\n";          }          ~GfG()          {              cout << "Inside Destructor\n";          }  };    int main()  {      int x = 0;      if (x==0)      {          GfG obj;      }      cout << "End of main\n";  } |

Output:

Inside Constructor

Inside Destructor

End of main

In the above program the object is declared inside the if block as non-static. So, the scope of variable is inside the if block only. So when the object is created the constructor is invoked and soon as the control of if block gets over the destructor is invoked as the scope of object is inside the if block only where it is declared.  
Let us now see the change in output if we declare the object as static.

|  |
| --- |
| // CPP program to illustrate  // class objects as static  #include<iostream>  using namespace std;    class GfG  {      int i = 0;        public:      GfG()      {          i = 0;          cout << "Inside Constructor\n";      }        ~GfG()      {          cout << "Inside Destructor\n";      }  };    int main()  {      int x = 0;      if (x==0)      {          static GfG obj;      }      cout << "End of main\n";  } |

Output:

Inside Constructor

End of main

Inside Destructor

You can clearly see the change in output. Now the destructor is invoked after the end of main. This happened because the scope of static object is throughout the lifetime of program.

 **Static functions in a class**: Just like the static data members or static variables inside the class, static member functions also does not depend on object of class. We are allowed to invoke a static member function using the object and the ‘.’ operator but it is recommended to invoke the static members using the class name and the scope resolution operator.  
**Static member functions are allowed to access only the static data members or other static member functions**, they can not access the non-static data members or member functions of the class.

|  |
| --- |
| // C++ program to demonstrate static  // member function in a class  #include<iostream>  using namespace std;    class GfG  {     public:        // static member function      static void printMsg()      {          cout<<"Welcome to GfG!";      }  };    // main function  int main()  {      // invoking a static member function      GfG::printMsg();  } |

Output:

Welcome to GfG!