1. What does the following do:

```
    void a  function(int *x)
 {
 x=new int;
 *x=12;
  }
 int main()
 {
 int v=10;
 afunction(&v);
 cout<<v;
 }
```

a) Outputs 12   b) Outputs 10   c) Outputs the address of v

2. How long does this loop run: for(int x=0; x=3; x++)

a) Never   b) Three times   c) Forever

3. Which uses less memory?

a)

```
struct astruct
 {
int x;
float y;
int v;
 };
```

b)

```
union aunion
```

```
 {
 int x;
 float v;
 };
```

c)

```
char array[10];
```

4. Evaluate:

```
 int fn(int v)
 {
 if(v==1 || v==0)
   return 1;
 if(v%2==0)
   return fn(v/2)+2;
 else
   return fn(v-1)+3;
 }
```

for fn(7);

a) 10 b) 11 c) 1

5. Which of the Standard C++ casts can be used to perform a "safe" downcast:

a) reinterpret_cast

b) dynamic_cast

c) static_cast

d) const_cast

6.

```
class professor {};

class teacher : public virtual professor {};

class researcher : public virtual professor {};

class myprofessor : public teacher, public researcher {};
```

Referring to the sample code above, if an object of class "myprofessor" were created, how many instances of professor will it contain?

a) 0 b) 1 c) 2 d) 3 e) 4

7.

string somestring ;

Which of the following choices will convert a standard C++ string object "somestring" to a C string?

a) Copy.somestring () ;

b) somestring.c_str ()

c) &somestring [1]

d) std::cstring (somestring)

e) (char *) somestring

8.

```
class basex

{

  int x;

public:

  void setx(int y) {x=y;}

};

class derived : basex {};
```

What is the access level for the member function "setx" in the class "derived" above?

a) private

b) local

c) global

d) public

e) protected

9.

```
class Alpha {
public:
    char data[10000];
    Alpha();
    ~Alpha();
};
class Beta {
public:
    Beta() { n = 0; }
    void FillData(Alpha a);

private:
  int n;
};
```

How do you make the above sample code more efficient?

a) If possible, make the constructor for Beta private to reduce the overhead of public constructors.

b) Change the return type in FillData to int to negate the implicit return conversion from "int" to "void".

c) Make the destructor for Alpha virtual.

d) Make the constructor for Alpha virtual.

e) Pass a const reference to Alpha in FillData

10.

```
class Foo {

    int x;

public:

    Foo(int I);

};
```

If a class does not have a copy constructor explicitly defined one will be implicitly defined for it. Referring to the sample code above, which one of the following declarations is the implicitly created copy constructor?

a) Foo(Foo *f);

b) Foo(Foo &f);

c) Foo(const Foo *f);

d) Foo(const Foo &f);

e) Foo(int);

11.

```
class HasStatic {

    static int I;

};
```

Referring to the sample code above, what is the appropriate method of defining the member variable "I", and assigning it the value 10, outside of the class declaration?

a) HasStatic I = 10;

b) int static I = 10;

c) static I(10);

d) static I = 10;

e) int HasStatic::I = 10;

12.

```
class X

{

private:

  int a;

protected:

  X(){cout<<"X constructor was called"<<endl;}

  ~X(){cout<<"X destructor was called"<<endl}

};
```

Referring to the code above, which one of the following statements regarding "X" is TRUE?

a) X is an abstract class.

b) Only subclasses of X may create X objects.

c) Instances of X cannot be created.

d) X objects can only be created using the default copy constructor.

e) Only friends can create instances of X objects.

13.

```
class Foo {

   const int x;

protected:

   Foo(int f);

   ~Foo();

};

Foo f;
```

Referring to the sample code above, why will the class declaration not compile?

a) The variable x is const.

b) The destructor is protected.

c) The destructor is not public.

d) The constructor is protected.

e) There is no default constructor.

14.

```
class Foo {

public:

    Foo(int i) { }

};

class Bar : virtual Foo {

public:

   Bar() { }

};



Bar b;
```

Referring to the above code, when the object 'b' is defined, a compiler error will occur. What action fixes the compiler error?

a) Adding a virtual destructor to the class Bar

b) Adding a constructor to Bar which takes an int parameter

c) Adding "Foo()" to the Bar constructor

d) Adding a copy constructor to the class Foo

e) Adding "Foo(0)" to the Bar::Bar initializer list

15. Which one of the following describes characteristics of "protected" inheritance?

a) The base class has access only to the public or protected members of the derived class.

b) The derived class has non-public, inheritable, access to all but the private members of the base class.

c) The derived class has access to all members of the base class.

d) The private members of the base class are visible within the derived class.

e) Public members of the derived class are privately accessible from the base class.

16. The "virtual" specifier in a member function enables which one of the following?

a) Monmorphism

b) Late binding

c) Metamorphism

d) Solomorphism

e) Inheritance

17.

```
class X

{

public:

  int x;

  static void f(int z);

};

void X::f(int y) {x=y;}
```

What is the error in the sample code above?

a) The class X does not have any protected members.

b) The static member function f() accesses the non-static z.

c) The static member function f() accesses the non-static x.

d) The member function f() must return a value.

e) The class X does not have any private members.

18.

```
template<class T, class X> class Obj {

   T my_t;

   X my_x;

public:

   Obj(T t, X x) : my_t(t), my_x(x) { }

};
```

Referring to the sample code above, which one of the following is a valid conversion operator for the type T?

a) T operator T () { return my_t; }

b) T operator(T) const { return my_t; }

c) operator(T) { return my_t; }

d) T operator T (const Obj &obj) { return obj.my_t; }

e) operator T () const { return my_t; }

19.

```
catch(exception &e)

{

   . . .

}
```

Referring to the sample code above, which one of the following lines of code produces a written description of the type of exception that "e" refers to?

a) cout << e.type();

b) cout << e.name();

c) cout << typeid(e).name();

d) cout << e.what();

e) cout << e;

20.

```
int f() {
    int I = 12;
    int &r = I;
    r += r / 4;
    int *p = &r;
    *p += r;
    return I;
}
```

Referring to the sample code above, what is the return value of the function "f()"?

a) 12

b) 15

c) 24

d) 17

e) 30

Answer key

1. b	2. c	3. b	4. b	5. b	6. b	7. b	8. a	9. e	10. d.
11. e	12. b	13. e	14. e	15. b	16. b	17. c	18. e	19. c	20. e

Explanation

6.b

Here professor will be called a virtual base class since teacher and researcher derive from it virtually. This is used in multiple inheritance as shown here. If professor was not inherited virtually then there would be 2 instances of professor in the object of myprofessor.

8. a

Table of Member Access Privileges

Access in Base Class	Base Class Inherited as	Access in Derived Class
Public Protected Private	Public	Public Protected No access[1]
Public Protected Private	Protected	Protected Protected No access[1]
Public Protected Private	Private	Private Private No access[1]

[1] Unless friend declarations within the base class explicitly grant access.

So, the highest member accessibility is defined by the way a class is inherited, if it is inherited privately, then the highest member accessibility will be private. Default inheritance is private

9.e

Since u r passing a reference hence a new array will not be created in memory, whereas if u pass by value, then an array of 10000 chars will be created. Passing by reference only creates an alias for the original parameter (i.e., it points to the original parameter) and is same as passing by address, the only difference is that it can be used like an object instead of as a pointer, i.e., if param is &a, then u will write a.member, whereas if param is *a then u will write a->member.

10.d
Copy constructor takes an arg of its own type which is passed by ref and which should not be changed hence it is const

11.e

Instances of X can be created only inside its subclasses.

13.e
If u don't specify a constructor for a class, then the compiler generates the default constructor for u, but if u specify a constructor apart from the default constructor, then u must give the default constructor also, but only if u r creating an object which uses the default constructor. i.e., the following code is perfectly fine

```
class Foo {

public:

    Foo(int f){}

    ~Foo(){}

};

Foo f(0);
```

But if u try to create an object of Foo like this

```
Foo f;
```

then this will give compiler error because there is no default constructor available.

16.b

Consider a base class B having a virtual function Foo, and a class D derived from this class also having a function Foo. Then when u create an object of a derived class, say dobj, and a ptr to base class, say pb, and point pb to dobj by saying pb = &dobj, and then call pb->Foo then the correct version of Foo will be called, i.e., the version in D will be called becz the object is of type D. this is called late binding, i.e., deferring the decision of calling which version of Foo until runtime since at compile time the type of object to which pb points may not be known.

```
class B {public: virtual void Foo(){cout<<"base";}};

class D : public B {public: virtual void Foo(){cout<<"derv";}};

D dobj;

B *pb;

pb = &dobj;

pb->Foo();
```

the output will be 'derv'.

If Foo was not declared virtual in base class B , then the output would have been 'base'.

18.e

this will be used for casting objects of type Obj to type T

```
class Myclass{};

template<class T, class X> class Obj {

    T my_t;

    X my_x;

public:

    Obj(T t, X x) : my_t(t), my_x(x) { }

  operator T () const { return my_t; }

  operator X () const { return my_x; }

};


void main()

{

              Myclass mt;

            Obj <int , Myclass > myobj(10,mt);

            int x = (int) myobj;

            cout << x;

              Myclass mobj = (Myclass) myobj;

}
```