This exam is pledged, and is closed book and closed notes. All answers are to be given on the provided answer sheet. For objective questions, if you find more than one answer is acceptable, choose the most encompassing answer. *YOU MUST HAND IN ALL COPIES OF THE EXAM AND ITS ANSWER SHEET.* This exam has 6 pages containing 45 questions.

Fill in the blank

1. A _____ is a collection of function interfaces, constant and variable object definitions, and class descriptions. It does not include implementation.
2. A function's computation is normally brought back as the _____ value.
3. The parameters in the invocation are called the _____ parameters.
4. Every function invocation creates an _____, which stores values of parameters and local objects.
5. The _____ is responsible for handling file inclusion directives.
6. When creating a new class type, two important initial steps are determining the _____ and the behaviors of the objects.
7. Member functions that return the value of an attribute of an object are called _____.
8. Member functions that set or change the value of an attribute of an object are called _____.
9. A _____ member function is one that cannot change its objects' attributes.
10. A _____ is a list of statements within curly braces.
11. A _____ object is an object defined within a statement block.
12. A _____ object is an object defined outside of any function interface or function body.
13. When a _____ parameter is used any modifications made to that parameter by the invoked function change the original object.
14. If an object is large, making a copy of it can be expensive in terms of time and space. Thus objects that are large or objects whose size is not known are often passed by _____.
15. A _____ modifier applied to a parameter declaration indicates that the function may not modify the object. If the function attempts to modify the object, the compiler reports an error.
16. Function _____ is when two or more functions have the same name.

True/false questions

17. Indices, also called referencing, are used to select an element of an array.
18. Default parameters must always occur after non-default parameters.
19. A programmer must put a **`return`** statement in a **`void`** function.
20. Auxiliary functions must use the object-dot notation to reference data members, but member functions need not.
21. Elements in an array always have the same type.
22. `a[3]` accesses the third element of the array `a`.

The following declarations are in effect for Questions 23 - 27.

```
class Widget {
   public:
      Widget ();
      int get() const;
   private:
      int dataValue;
      void set(int v);
};

Widget t;
```

23. The following statement is legal for function `main()`.

```
t.dataValue = 3;
```

24. The following statement is legal for function `main()`.

```
int k = t.get();
```

25. The following statement is legal for function `main()`.

```
t.set(3);
```

26. The following statement is legal for `Widget` member function `get()`.

```
dataValue = 3;
```

27. The following statement is legal in a `Widget` member function.

```
int k = t.get();
```

The following code fragment is in effect for Questions 28 - 31.

```
int Coffee(bool cream, sugar) {
   char cups[9];
   for (int i = 0; i <= 9; i) {
      if (cream && sugar) {
         cups[i] = "5";
      }
   }
}
```

28. The parameter list correctly defines that there are two **`bool`** value parameters.

29. The definition of `cups` defines an array of nine **`char`** elements.

30. The assignment to `cups[i]` within the **`if`** statement is correct.

31. The **`for`** loop iterates ten times.

32. What is the output of the following program?

```
#include <iostream>
using namespace std;
int counter = 0;
void f() {
   ++counter;
}
void g() {
   f();
   f();
}
void h() {
   f();
   g();
}
int main() {
   g();
   cout << ++counter << endl;
   h();
   cout << ++counter << endl;
   return 0;
}
```

33. Consider the following function `scramble()`:

```
void scramble(int i, int &j, int &k) {
   i = 10;
   j = 20;
   k = 30;
   return;
}
```

What is the output of the following program fragment?

```
#include <iostream>
using namespace std;
int main() {
   int i = 1;
   int j = 2;
   int k = 3;
   scramble(k, j, i);
   cout << i << " " << j << " " << k << endl;
   return 0;
}
```

34. What is the output of the following program?

```
#include <iostream>
using namespace std;
void f(int i, int j) {
   int temp;
   temp = i;
   i = j;
   j = temp;
   cout << i << " " << j << endl;
   return;
}
int main() {
   int a = 10;
   int b = 20;
   f(a, b);
   f(b, a);
   return 0;
}
```

35. What does the following correct program print out?

```
#include <iostream>
using namespace std;
const int LoopBound = 5;
int i;
int f(int a) {
   for (i = 0; i <= LoopBound; ++i) {
      if ( i <= a ) {
         cout << i << "  ";
      }
   }
   cout << endl;
}

int main() {
   for (i = 0; i < 2; ++i) {
      f(i);
   }
   return 0;
}
```

36. What is the output of the following program fragment?

```
int a[5];
a[0] = 0;
for (int i = 1; i <= 4; ++i) {
   a[i] = a[i-1] + i;
}
cout << a[3] << endl;
```

The following definitions are in effect for questions 37– 40. Member functions have the purposes that their names suggest.

```
class Rational {
   public: // member functions and operator
      Rational();
      Rational(const int numer, const int denom);
      bool Equal(const Rational &r) const;
      void IncrementByOne(const Rational &r);
      Rational Reciprocal() const;
   protected:
      int GetNumerator() const;
      int GetDenominator() const;
      void SetNumerator(const int numer);
      void SetDenominator(const int denom);
   private:
      int Numerator;
      int Denominator;
};
```

37. The **const** in the prototypes of functions `GetNumerator()` and `GetDenominator()` in class `Rational` indicates that

    (a) The functions will return a constant value.

    (b) The functions will not modify their parameters.

    (c) The functions will not change the object.

    (d) None of the above.

38. Fill in the blank using only objects `a`, `b`, `c`, and `d` to complete the following member function `Equal()` so that it returns **`true`** if and only if the invoking `Rational` is mathematically equivalent to its parameter (e.g., the rational 1/2 is equal to the rational 2/4).

```
bool Rational::Equal(const Rational &r) {
   int a = GetNumerator();
   int b = GetDenominator();
   int c = r.GetNumerator();
   int d = r.GetDenominator();
   return ____________________________ ;
}
```

39. Complete the definition of the auxiliary operator `!=` that returns **`true`** if and only if the rational numbers represented by its two operands are not equivalent.

40. Complete the definition of the member function `Reciprocal()` that returns the reciprocal of the invoking `Rational`. For example if the invoking `Rational` was equivalent to *a*/*b*, then the function returns a `Rational` equal to *b*/*a*. Do not use division in completing your answer.

41. What is the output of the following program?

```
#include <iostream>
using namespace std;
void f(int i, int j) {
   i = i + j;
   j = j + i;
   return;
}
int main() {
   int i = 10;
   int j = 20;
   f(50, j);
   f(i, 50);
   cout << i << " " << j << endl;
   return 0;
}
```

Coding

42. Prototype a function `IsMathSymbol()` that takes a character value `c` and returns a Boolean value.

43. Write a single statement function body for a **`float`** function `RectangleArea()` that computes and returns the area of a rectangle using its two **`float`** formal parameters `h` and `w`, where `h` is the height and `w` is the width of the rectangle.

44. Consider the following program:

```
#include <iostream>
// your prototype goes here
using namespace std;
int main() {
   int x = 1;
   f(x);
   cout << "x is " << x << endl;
   return 0;
}
```

When this program runs, the output is

```
x is 2
```

Write the function prototype for **`void`** function `f()`.

45. Suppose drawing outside the left border of a window causes an error and your program to fail. Let `W` be a `SimpleWindow` that is `m` cm wide and `n` cm high. Give code that successfully draws in `W` the visible portion of 1 cm by 1 cm `RectangleShape` `R` with center coordinates (`cx`, `cy`).