1 Short Answer

1. What is the difference between a protected class member and a private class member?

   Solution: protected members can be seen by derived classes whereas private members cannot.

2. Which constructor is called first, that of the derived class or the base class?

   Solution: The base class constructor is called first.

3. When does static binding take place? When does dynamic binding take place?

   Solution: Static binding takes place at compile time and dynamic binding takes place at run time.

4. What is an abstract base class? How is it crated? What distinguishes it from a non-abstract base class?

   Solution: An abstract base class is one that contains at least one purely virtual function. These are defined with an $= 0$ at the end of their specification. Abstract classes cannot be instantiated.

5. Write the first line of the declaration for a Poodle class. The class should be derived from the Dog class with public base class access.

   Solution: `class Poodle: public Dog`

6. Write the first line of the declaration for a SoundSystem class. Use multiple inheritance to base the class on the MP3player class, the Tuner class, and the CDPlayer class. Use public base class access in all cases.

   Solution: `class SoundSystem: public MP3player, public Tuner, public CDPlayer`

7. What is a container class?

   Solution: A container is a class that stores data and organizes it in some fashion.

8. What is an iterator? Write the definition of one for a vector of doubles.

   Solution: An iterator is an object that behaves like a pointer. It is used to access the individual data elements in a container.

   ```
   vector<double>::iterator iter;
   ```

9. Write a function named `fct` that takes two integer parameters a and b. If a is less than b the function should throw an exception of an integer error code of 123. Also write the segment of code in the main that will call this function and catch the exception. Output from these segments of code is below.

   ```
   Enter two numbers: 5 3
   The sum is 8

   Enter two numbers: 1 4
   Error 123
   ```

   Solution:

   ```
   int fct(int a, int b) {
     if (a < b) {
       throw 123;
     }

     return a + b;
   }
   ```

   ```
   int a, b;
   cout << "Enter two numbers: ";
   cin >> a >> b;

   try {
     int sum = fct(a, b);
     cout << "The sum is " << sum << endl;
   } catch (int &e) {
     cout << "Error " << e << endl;
   }
   ```

10. Write a templated function called `print` that brings in a vector of any type and prints its contents on a single horizontal line with a space between the entries. Have it do a new line at the very end and do the printing in a range-based loop.

    Solution:

```
template <class T> void print(vector<T> const &vect) {
  for (T v : vect)
    cout << v << " ";
  cout << endl;
}
```

11. Write a templated bubble, insertion, or selection sort (just one of these) that will sort an array (regular array, not STL) of any data type that has the relational operators < and > defined as well as assignment.

    Solution:

```
template <class T> void bubble(T A[], int size) {
  for (int i = 0; i < size - 1; i++) {
    for (int j = 0; j < size - i - 1; j++) {
      if (A[j] > A[j + 1]) {
        T temp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = temp;
      }
    }
  }
}

template <class T> void insertion(T A[], int size) {
  for (int i = 0; i < size; i++) {
    int j = 0;
    T val = A[i];
    for (j = i; j > 0 && A[j - 1] > val; j--)
      A[j] = A[j - 1];

    A[j] = val;
  }
}

template <class T> void selection(T A[], int size) {
  int minindex;

  for (int i = 0; i < size; i++) {
    minindex = i;
    for (int j = i; j < size; j++)
      if (A[j] < A[minindex])
        minindex = j;

    T val = A[i];
    A[i] = A[minindex];
    A[minindex] = val;
  }
}
```

2 Coding Exercise #1

This exercise is to write an inheritance structure for triangles. Recall from geometry that an Isosceles triangle is one with two equal sides and that an Equilateral triangle is one where all three sides are of equal length.

Write a Triangle class that has a default constructor and one that takes in the three sides as parameters. This class should store the lengths of the three sides of the triangle. In addition it should have the following functions.

- Area: this function will calculate and return the area of the triangle. Recall that if the sides of the triangle are a, b, and c then let p be the semi-perimeter $p = (a + b + c)/2$ then the area is

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

- Sides: this prints to the screen the lengths of the sides.
- Draw: this prints "Draw Triangle" to the screen.

Write an Isosceles class that inherits off the Triangle class. There must be a default constructor and one that takes two parameters, the first is the double sides of equal length and the third is the length of the last side. That is, `Isosceles(3, 5)` is a triangle with side lengths 3, 3, and 5. This class must also be able to call the functions Area, Sides, and Draw, but in this case the Draw command will print to the screen, "Draw Isosceles Triangle".

Write an Equilateral class that inherits off the Isosceles class. There must be a default constructor and one that takes one parameter, the length of all the sides. That is, `Equilateral(7)` is a triangle with side lengths 7, 7, and 7. This class must also be able to call the functions Area, Sides, and Draw, but in this case the Draw command will print to the screen, "Draw Equilateral Triangle".

Write the specifications and implementations for each of the three classes below. **Each of the following pages is for each separate file, the filename to be written is at the top of the page.**

There is to be no inline code. There is a block of sample code below and its output. Read this very closely, your class structures are to produce exactly the same output to this sample code.

Sample Code

```
Triangle *tris[5];
tris[0] = new Triangle(3, 4, 5);
tris[1] = new Isosceles(3, 5);
tris[2] = new Isosceles(4, 5);
tris[3] = new Equilateral(7);
tris[4] = new Triangle(7, 5, 3);

for (int i = 0; i < 5; i++) {
    tris[i]->Draw();
    cout << tris[i]->Area() << endl;
    tris[i]->Sides();
    cout << endl;
}
```

Output

```
Draw Triangle
6
Side Lengths: 3 4 5

Draw Isosceles Triangle
4.14578
Side Lengths: 3 3 5

Draw Isosceles Triangle
7.80625
Side Lengths: 4 4 5

Draw Equilateral Triangle
21.2176
Side Lengths: 7 7 7

Draw Triangle
6.49519
Side Lengths: 7 5 3
```

Solution:

```
#ifndef TRIANGLE_H_
#define TRIANGLE_H_

class Triangle {
protected:
    double a;
    double b;
    double c;

public:
    Triangle(double sideA = 0, double
        sideB = 0, double sideC = 0);
    double Area();
    void Sides();
    virtual void Draw();
};

#endif /* TRIANGLE_H_ */
```

```
#include <cmath>
#include <iostream>

#include "Triangle.h"

using namespace std;

Triangle::Triangle(double sideA, double
    sideB, double sideC) {
    a = sideA;
    b = sideB;
    c = sideC;
}

double Triangle::Area() {
    double p = (a + b + c) / 2;
    double area = sqrt(p * (p - a) * (p -
        b) * (p - c));
    return area;
}

void Triangle::Sides() {
    cout << "Side Lengths: " << a << " "
        << b << " " << c << endl;
}

void Triangle::Draw() {
    cout << "Draw Triangle" << endl;
}
```

```
#ifndef ISOSCELES_H_
#define ISOSCELES_H_

#include "Triangle.h"

class Isosceles: public Triangle {
public:
    Isosceles(double d = 0, double t = 0);
    void Draw();
};

#endif /* ISOSCELES_H_ */
```

```
#include <iostream>

#include "Isosceles.h"

using namespace std;

Isosceles::Isosceles(double d, double t):
    Triangle(d, d, t) {
}

void Isosceles::Draw() {
    cout << "Draw Isosceles Triangle" <<
        endl;
}
```

```
#ifndef EQUILATERAL_H_
#define EQUILATERAL_H_

#include "Isosceles.h"

class Equilateral: public Isosceles {
public:
    Equilateral(double a = 0);
    void Draw();
};

#endif /* EQUILATERAL_H_ */
```

```
#include <iostream>

#include "Equilateral.h"

using namespace std;

Equilateral::Equilateral(double a):
    Isosceles(a, a) {
}

void Equilateral::Draw() {
    cout << "Draw Equilateral Triangle" <<
        endl;
}
```

3 Coding Exercise #2

Write a templated class named `Tarray` that stores a dynamic one-dimensional array (normal array, not an STL array) of the parameter type T. It also stores the size of the array and a default value for populating the array. The class should have the following functions. Example main and output is below. There is to be no inline code. **Each of the following pages is for the code of the specified functions.**

- Constructor that brings in parameters for the array size and default value. No default vales here and no default constructor is to be written.
- Destructor, copy constructor and overloaded assignment operator.
- A size function that returns the size if the array.
- Min and Max functions that return the minimum and maximum values in the array.
- Set function that takes in an array index and values and places that value in the array. If the index is out of bounds of the allocated array memory then the function should throw an exception string of `Index out of bounds.`.
- Get function that returns the element of the array at the specified (parameter) index. If the index is out of bounds of the allocated array memory then the function should throw an exception string of `Index out of bounds.`.
- Overloaded [ ] operator. If the index is out of bounds of the allocated array memory then the function should throw an exception string of `Index out of bounds.`.
- A print function that prints the array elements with a comma between the entries.

```
#include "Tarray.h"
#include <iostream>
using namespace std;

int main() {
  Tarray<int> A(10, 0);
  A.Print();
  for (int i = 0; i < A.size(); i++)
    A[i] = (3 * i + 1) % A.size();
  A.Print();
  cout << A.Max() << " " << A.Min() <<
      endl;
  cout << endl;
  Tarray<int> B(5, 3);
  Tarray<int> C(20, 1);
  B.Print();
  C.Print();
  cout << endl;
  B = C = A;
  A.Print();
  B.Print();
  C.Print();
  cout << endl;
  B[2] = B[4] = B[7] = -5;
  C[1] = C[2] = C[3] = -1;
  A.Print();
  B.Print();
  C.Print();
  cout << endl;
  Tarray<int> D(A);
  A.Print();
  D.Print();
  D.set(5, -2);
  D.set(1, -5);
  D.set(7, 0);
  D.Print();
  for (int i = 0; i < A.size(); i++)
    cout << A.get(i) << " ";
  cout << endl << endl;
  Tarray<string> S(5, "Jack");
  S[0] = "Sue";
  S[1] = "James";
  S[2] = "George";
  S[3] = "Don";
  S.Print();
  cout << S.Max() << " " << S.Min() <<
      endl;
  return 0;
}
```

Output:

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0
1, 4, 7, 0, 3, 6, 9, 2, 5, 8
9 0

3, 3, 3, 3, 3
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

1, 4, 7, 0, 3, 6, 9, 2, 5, 8
1, 4, 7, 0, 3, 6, 9, 2, 5, 8
1, 4, 7, 0, 3, 6, 9, 2, 5, 8

1, 4, 7, 0, 3, 6, 9, 2, 5, 8
1, 4, -5, 0, -5, 6, 9, -5, 5, 8
1, -1, -1, -1, 3, 6, 9, 2, 5, 8

1, 4, 7, 0, 3, 6, 9, 2, 5, 8
1, 4, 7, 0, 3, 6, 9, 2, 5, 8
1, -5, 7, 0, 3, -2, 9, 0, 5, 8
1 4 7 0 3 6 9 2 5 8

Sue, James, George, Don, Jack
Sue Don
```

Solution:

```
#ifndef TARRAY_H_
#define TARRAY_H_

#include <iostream>

using namespace std;

template <class T> class Tarray {
private:
  T *A;
  int sz;
  T def;

public:
  Tarray(int size, T defval);
  Tarray(const Tarray &obj);
  virtual ~Tarray();
  int size();

  void set(int, T);
  T get(int pos);
  T Max();
  T Min();

  const Tarray operator=(const Tarray &right);
  T &operator[](int);
  void Print();
};

template <class T> Tarray<T>::Tarray(int size, T defval) {
  sz = size;
  def = defval;
  A = new T[sz];
  for (int i = 0; i < sz; i++)
    A[i] = def;
}

template <class T> Tarray<T>::Tarray(const Tarray &obj) {
  sz = obj.sz;
  def = obj.def;
  A = new T[sz];
  for (int i = 0; i < sz; i++)
    A[i] = obj.A[i];
}

template <class T> const Tarray<T> Tarray<T>::operator=(const Tarray &right) {
  if (this == &right)
    return *this;

  delete[] A;
  sz = right.sz;
  def = right.def;
  A = new T[sz];
  for (int i = 0; i < sz; i++)
    A[i] = right.A[i];

  return *this;
}

template <class T> Tarray<T>::~Tarray() { delete[] A; }

template <class T> int Tarray<T>::size() { return sz; }

template <class T> void Tarray<T>::set(int pos, T item) {
  if (pos < 0 || pos > sz - 1)
    throw string("Index out of bounds.");
  A[pos] = item;
}

template <class T> T Tarray<T>::get(int pos) {
```

```
  if (pos < 0 || pos >= sz)
    throw string("Index out of bounds.");
  return A[pos];
}

template <class T> T Tarray<T>::Max() {
  T max = A[0];
  for (int i = 0; i < sz; i++)
    if (A[i] > max)
      max = A[i];

  return max;
}

template <class T> T Tarray<T>::Min() {
  T min = A[0];
  for (int i = 0; i < sz; i++)
    if (A[i] < min)
      min = A[i];

  return min;
}

template <class T> T &Tarray<T>::operator[](int pos) {
  if (pos < 0 || pos >= sz)
    throw string("Index out of bounds.");
  return A[pos];
}

template <class T> void Tarray<T>::Print() {
  for (int i = 0; i < sz; i++)
    if (i < sz - 1)
      cout << A[i] << ", ";
    else
      cout << A[i] << endl;
}

#endif /* TARRAY_H_ */
```