Homework #2

Some questions in this and future homework assignments will require you to submit multiple files as the answer to a single question. You can do this by packing those files into one single ZIP file, using the `zip` Linux command on the COE machines, and then importing the created ZIP file on your local machine. For example, the following command creates a ZIP file named `question.zip`, which includes files `file1.cc` and `file2.cc`:

```
$ zip question.zip file1.cc file2.cc
```

Question 1 (3 pt.)

Consider the following iterative implementation of a binary search algorithm, where *A* represents an input sorted vector, and *x* is the element being searched:

```
BINARY-SEARCH(A, x)
    left = 0
    right = A.length – 1
    while left ≤ right
        n = right – left + 1
        mid = left + n / 2
        if A[mid] = x
            return mid
        if A[mid] < x
            left = mid + 1
        else
            right = mid – 1
    return -1
```

a) (1 pt.) Calculate the temporal cost of the algorithm in the best case T_{best} when element *x* is located right in the middle of *A*, and in the worst case T_{worst} when element *x* is not present in the vector at all. Give the cost functions in number of lines of pseudo-code being executed as a function of the size of the input vector *n*. For simplicity, you can restrict your analysis to those cases where *n* is a power of 2.

b) (2 pt.) Prove that the asymptotic cost of an iterative binary search in the worst case is $T_{worst} = \Theta(\log_2 n)$. This process involves proving that function $\log_2 n$ is both an asymptotic upper bound $T_{worst} = O(\log_2 n)$ of the cost function, as well as an asymptotic lower bound $T_{worst} = \Omega(\log_2 n)$. Use the formal definitions presented in class for the asymptotic bounds.

Question 2 (2 pt.)

The *shapes* project presented in class illustrates multiple key features of the C++ programming language, as listed next. Explain how each of these features is manifested in a particular position of the code.

a) Data protection (given as an example below).
b) Class constructors and destructors.
c) Class instantiation.
d) Virtual functions.
e) Pure virtual functions.
f) Abstract classes.
g) Polymorphism.
h) Include guards.

Example: The data protection feature of C++ is manifested, for example, in the declaration of class `Rectangle` in file `Rectangle.h` using access specifiers `public` and `private`. Fields listed under the `private` section are accessible only within the class, while functions listed under the `public` section are accessible from any other class or function.

Question 3 (2 pt.)

Modify the declaration of class `Shape` in file `Shape.h` by removing keyword `virtual` from its destructor `~Shape(void)`. What is the difference in the output of the program? What is this difference due to? Which version do you think is preferable? Explain what is the code being invoked in each case. You do not need to submit any source code for this question.

Question 4 (3 pt.)

Create a new class `Square` as a child of class `Rectangle`. Modify the `main()` function to instantiate one more shape object of type `Square`, add it to array `shapes`, and print its area within the `for` loop. The call to instantiate a `Square` object should take only one `float` value as an argument, specifying the length of a side of the square. Show the code for files `Square.cc`, `Square.h`, and `main.cc`.

Create a ZIP file with the complete source code for this question, and upload it in a file named `q4.zip`. Include all source files in the *Shapes* project, even those that were not modified, and omit any object or executable files.

Question 5 (open-ended, +2 pt.)

Extend class `Shape` with a virtual function `void Shape::Draw(void)` that draws the shape on the screen. This function can be optionally overridden by any child class of `Shape`. For example, a rectangle of width 8 and height 4 would draw the following figure:

```
Shape 'rect1'
Width = 8
Height = 4

********
*      *
*      *
********
```

You can decide for how many sub-classes of Shape you want to implement this functionality. Create a ZIP file with the complete source code for this question, and upload it in a file named `q5.zip`. Include all source files in the *Shapes* project, even those that were not modified, and omit any object or executable files.