**Question set 3**

**Chapter 2:**

An \_\_**abstract data type** is a data type whose properties (domain and operations) are specified independently of any particular implementation.

We refer to the set of all possible values (the domain) of an encapsulated data object plus the specifications of the operations that are provided to create and manipulate the data as an \_\_\_\_**ADT**\_\_\_\_\_\_.

A \_**data structure**\_ is a collection of data elements whose organization is characterized by accessing operations that are used to store and retrieve the individual data elements; the implementation of the composite data members in an abstract data type.

A \_\_**data structure**\_\_\_\_ is defined by the logical arrangement of data elements combined with the set of operations we need to access the elements.

A \_\_**composite**\_\_ is a data type that allows a collection of values to be associated with an object of that type.

A colloquial definition of a \_\_**class**\_\_\_\_\_\_\_ is a combination of data items and the functions used to manipulate them. class

A \_\_**1-D**\_\_\_ array is a structured composite data type made up of a finite, fixed-size collection of ordered homogenous elements to which direct access is available.

In an array, \_\_**finite**\_\_ indicates that a last element is identifiable.

\_**Fixed size**\_\_ means the size of the array must be known in advance.

An \_\_**Ordered**\_\_ array means there is a first element, a second element and so on. The relative position of the elements is ordered, not necessarily the values stored there.

The elements in an array are physically \_**homogeneous**\_. That is, they are all of the same data type.

The component selection mechanism of an array is \_\_**direct access**\_ – that means we can access any element directly without first accessing the preceding elements.

In C++ arrays can only be \_\_**reference**\_\_ parameters, it’s not possible to pass an array by value.

A \_\_\_**1-D array**\_\_ is the natural structure for the storage of lists of like data elements. An array is a data structure that contains a group of \_**elements**\_\_\_ that are all of the same data type, such as an integer or string.

A two-dimensional array is commonly referred to as a \_**matrix**\_\_.

A \_\_**2-d**\_\_ array is a structured composite data type made up of a finite, fixed-size collection of homogeneous elements ordered in two dimensions and accessed by direct access.

A \_**client**\_ is software that declares and manipulates instances of objects of a particular class.

The attributes (variables) and the methods (functions) of a class are called \_\_**members**\_\_.

Although the class specification (declarations) and the implementation (definitions) can reside in the same file, the two parts of a class are usually separated into two files. The specification goes into a header file and has an \_**.h**\_ extension.

Although the class specification (declarations) and the implementation (definitions) can reside in the same file, the two parts of a class are usually separated into two files. The implementation goes into a file with the same name as the header and has an \_\_**.cpp**\_\_ extension.

The rules of C++ that governs what, where and when are called \_**scope rules**\_.

\_**class scope**\_ refers to identifiers (variables) declared within a class declaration.

\_**local scope**\_ is the scope of an identifier (variable) declared within a block (statements enclosed within { } ).

\_**global scope**\_ is the scope of an identifier (variable) declared outside all functions and classes.

All identifiers declared within a class are local to the class. This is called \_**class scope**\_.

The name of a function that is not a member of a class has global scope. Once a global function name has been declared, any subsequent function can call it. This is called \_\_\_**global scope**\_.

Given the existence of a class DateType, a client of the class DateType must have an \_**#include**\_\_ directive for the header file of the class.

System-supplied header files are enclosed in angle brackets ( <iostream> ) where user-defined header files are enclosed in \_**double quotes**\_\_\_\_\_\_\_\_\_.

Only the member functions can access the data members, so we must associate the class name with the function definitions. We do so by inserting the class name before the function name, separated by the \_**scope resolution operator (:)**\_\_.

Member functions of a class are invoked in the same way that data members of a struct are accessed – using \_**dot notation**\_\_.

An object is an \_\_**instance**\_\_\_\_\_\_\_\_ of a class type.

\_**overloading**\_ means giving the same name to several different functions (or using the same operator symbol for different operations) in a class.

The time at which a name or symbol is bound to the appropriate code is called \_**binding time**\_\_\_.

The compile-time determination of which implementation of an operation is appropriate is called \_\_**static binding**\_\_\_.

The run-time determination of which implementation of an operation is appropriate is called \_**dynamic binding**\_\_\_.

**From Pages 134-137**

Lists are very useful abstract data types (ADTs). They are members of a general category of ADTs called \_**containers**\_\_, whose purpose is to hold other objects.

Lists are provided in the \_\_**standard template library**\_\_\_\_.

A \_\_**list**\_ is a homogenous collection of elements with a linear relationship between elements.

The number of items in the list, which we will call the \_**length**\_ of the list, is a property of a list.

Lists can be \_**unsorted**\_ - their elements are placed in no particular order.

Lists can be \_**sorted**\_ - their elements are placed in a particular order.

When the elements in a sorted list are of a composite type, their logical (and often physical) order is determined by one of the members of the structure, called the \_**key**\_ member.

In general, we can group the instance methods of a class into 5 categories. They are \_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_. Constructors, transformers, observers, iterators, destructors

A \_**constructor**\_ creates an instance of a data type.

\_**transformers**\_ are operations that change the structure in some way. They may make the structure empty, put an item into the structure or remove a specific item from a structure.

\_\_**observers**\_\_ ask true/false questions about the data type, select or access a particular item or return a property about a structure.

\_\_**iterators**\_\_\_ are used with composite data types to allow the user to process an entire structure component by component (or element by element).

\_**transformer**\_ alter the data fields of an object.

\_**observers**\_ read or otherwise get information about the data fields.

\_**constructors**\_ creates data for the ADT.

\_**iterators**\_ provide the ability to move through data fields one at a time.

A \_\_**destructor**\_ is executed when an object of it’s class goes out of scope or when the delete expression is applied to a pointer to the object of that class.

A \_\_**destructor** will have the exact same name as the class prefixed with a tilde (~). It can never take a value nor can it take any parameters. They can be used for releasing resources before coming out of a program like closing files, releasing memory, releasing printers, etc.

A \_**function prototype**\_ is a declaration.

A function heading with a body is a \_**function defintion**\_.

**Chapter 5:**



In the picture above, these are examples of \_**stack**\_.

At the logical level, a \_**stack**\_ is an ordered group of homogenous items or elements.

In a stack, the removal of existing items and the addition of new items can only take place at the \_**top**\_ of the stack.

A stack may be considered an \_**ordered**\_ group of items because elements occur in a particular sequence organized by how long they have been in the stack.

Because elements are added and removed only from the top of the stack, the last element to be added is the first to be removed. This stack behavior is referred to as \_**LIFO**\_.

The operation that adds an element to the top of the stack is usually called a \_\_**push**\_\_.

The operation that removes an element from a stack is called a \_**pop**\_.

If we need to examine the item at the top of the stack without removing the item, we can use a separate operation called \_**top**\_.

We must be able to tell whether a stack contains any elements before we pop it, so we need a Boolean operation called \_**IsEmpty**\_.

You may need to see if a stack is full before pushing. We call this operation \_\_**IsFull\_**.

A stack and an \_\_**unsorted list**\_ are two different Abstract Data Types (ADTs).

With a stack and an unsorted list can use the same \_**implementation strategy**\_ for both.

If the stack is full when we invoke Push, the resulting condition is called \_\_\_\_\_\_\_\_\_\_. A stack, queue or list can also be described as an \_\_\_\_\_\_\_\_\_\_. abstract data structure

Given an abstract data structure where an item is added at one end and removed from another. This structure is called a \_**FIFO**\_ queue.

A \_**queue**\_\_\_ is an ordered, homogenous group of elements in which one element is added at the rear and removed from the front.

To add elements to the queue, we access the rear of the queue. To remove elements we access the front of the queue. The middle elements are logically \_\_**inaccessible**\_.

Like a stack, a queue is a \_**holding structure**\_ for data we will use later.

When we add an element to the rear of a queue, this operation is called \_**enqueue**\_\_.

When we remove an element from the front of a queue, this operation is called \_**dequeue**\_.

(**True** or False) The stack operations *push* and *pop* are standard names.

(True or **False**) The queue operations *enqueue* and *dequeue* are standard names.

Computer systems must often provide a “Holding area” for messages between two processes, two programs or even two systems. This holding area, usually called \_**buffer**\_ is often implemented as a FIFO queue.