Summary Sheet

Arrays 10%

- Must be contiguous
- In many languages, they are a fixed number of slots (size) and set at compile time.
- In general it should be assumed that arrays cannot be expanded in size at run time. (There are exceptions in some languages)
- Each element must be the same data type (can include structures and classes in each slot)
- Arrays can be searched sequentially.
- Insertion of a new object in a sorted list may require the movement of substantial parts of the array. On average, ½ of the array will have to be moved to insert the new object.
- If the array is sorted, a binary search algorithm can be applied to maximize search speed.
- For each dimension in an array, there is an additional counter and an additional loop.

See this link for arrays http://www.ccpedagogy.com/cpp/1dimarrays.pdf

Pointers 10%

- They use a system generated address to point to where the object is in memory.
- The object pointed to can be standard data types, structures, classes or arrays as an example.
- The size of the pointer is determined by the operating system based on the amount of memory in the host computer.
- The size of the object pointed to by the pointer is determined by the size of the data type or the size of the user defined object.
- They are useful in linked lists and binary trees.
- Pointer arithmetic must be an integer. p++ or p—is typical.
- Declaration of a variable which is a pointer typically looks like :
  Data type * pointername
  int *ptr;
- Reference to the object that a pointer points to is :
  cout << *ptr; // Prints the value of the integer;
- Reference to the a field in the object that a pointer points to is :
  cout << ptr->name; // Prints the text that is in the field ***name*** of the object pointed to by ptr;
- Reference to the address of the object that is in the pointer :
  cout << ptr; // Prints the address of the integer;

See this link for pointers http://www.ccpedagogy.com/cpp/pointers%20to%20Arrays.pdf

Linked Lists (Pointer based system) 5%

- Linked lists must have a head to the list.
- Each node must have a pointer to the next item in the list.
- Access is sequential.
- Elements are added or deleted as needed during run time.
- Insertion of a new node in a sorted list requires only the changing of two pointers. Compared to arrays, this results in minimal data movement.
- The size of the linked list is expandable to the individual memory capabilities. (Flexible is a key concept)
- Overhead is that each node must have a pointer to the next object. Not needed in arrays since they are contiguous.

See this link for linked lists http://www.ccpedagogy.com/cpp/linked%20lists.pdf

Binary Trees (Pointer based system) 5%

- Binary trees must have a root (similar to the head in singly linked list) to the list.
- Each node must have a pointer to the left and right child in the list.
- Elements are added or deleted as needed during run time.
- Insertion of a new node in a sorted list requires only the changing of a single pointer on the end of the tree leaf.
- Recursion is part of the process.
- Assuming the tree is properly balanced, a binary search algorithm can be applied to the tree which can duplicate the speed of an binary search on an array.
- The size of the binary tree is expandable at run time to the individual memory capabilities of the device. (Flexible is a key concept)
- Overhead is that each node must have two pointers to the left and right child. Not needed in arrays since they are contiguous.

See this link for binary trees http://www.ccpedagogy.com/cpp/Binary%20Trees.pdf

Structures 10%

- A user defined data type which typically composed of fields or characteristics of different types which is used to describe an object.
- Also known as records in other applications.
- The initial step to object based programming
- Can have arrays of structures or arrays in structures.
- Can have nested structures, arrays, pointers and classes.
- When copying a structure, pointers and arrays (since they are pointers) in the structure must initialized or copied in a separate operation.
- Can be referenced, passed as parameters or modified by outside functions.
- Structures can be thought of as a subset of classes.

See these links for structures http://www.ccpedagogy.com/cpp/Struct.pdf http://www.ccpedagogy.com/cpp/SortStructs.pdf

Classes 30%

- A user defined data type which typically composed of fields or characteristics of different types which is used to describe an object.
- Data can be encapsulated by the use of private or protected status.
    - Private or protected variables/functions can only be referenced by functions which are part of the class. This feature allows the class to restrict changes or access from outside the class, which is a concept called encapsulation.
    - An exception to private feature is that the class may choose to designate a function or other class as a friend. The "friend" status conveyed on the designated function or class is not reciprocal. See http://www.ccpedagogy.com/cpp/Classes%20Friend%20functions.pdf
    - Public variables or functions can be called, referenced or modified by outside the class.
    - Encapsulated data allows the class to ensure that the proper type and range of data is part of the class. The data may still be incorrect as discussed in class regarding test scores (0-100 is a valid range but 7 instead of 70 may be incorrect).
- Can have arrays of structures or arrays in classes.
- Can have nested structures, arrays, pointers and classes.
- When copying a class, pointers and arrays in the structure must copied in a separate operation.
- Constructors inside the class can be invoked to assign standard values to part or all the fields in a class. There typically should be a default constructor with optional constructors.
- Destructors are used to return the memory of the object to the operating system. Only one destructor should be present and it is often just a noise word which does nothing.

See these links for classes

Structures to Classes http://www.ccpedagogy.com/cpp/classes2.pdf

Classes private/public http://www.ccpedagogy.com/cpp/classes3.pdf

Rock/Paper/Scissors http://www.ccpedagogy.com/cpp/Rock.pdf

Overloading Functions/Constructors/Operators 5%

- Functions can be overloaded by passing different objects to the same function name (see rock/paper/scissors example at this link http://www.ccpedagogy.com/cpp/Rock.pdf)
- Functions can be overloaded by changing the order or number of parameters to the same function name.
- Constructors can be overloaded by changing the order or number of parameters to the same class. (See http://www.ccpedagogy.com/cpp/Constructors.pdf)
- Operators can be overloaded. This technique is essence substitutes an operator symbol in place of a function name. Could be useful in situations which programmers in different countries where use of terms in different languages may be confusing in coordination between programmers.
  See http://www.ccpedagogy.com/cpp/operatorsFriendsThis.pdf

Data files 10%

- Datafiles can be text or binary.
- Text files generally must be input (read from) or output (write to) since the size of character data in particular may variable in size. They are generally not random access files.
- Binary files are read and written to and from in a machine code format. The use of structured (using structures or classes combined with arrays) allow the files to be random access and they can be read and written since each object is a standard size.

  See http://www.ccpedagogy.com/cpp/Binary%20files.pdf

Strings/ Cstrings 5%

- C strings are arrays of characters typically found in C programming. The techniques of array processing in normal arrays also apply to C strings.
- C strings must be declared an array of character with a fixed size.
- C Strings have a library of string functions which are similar to strings
- Strings in C++ (not in C) do not have to declared to be a fixed size;
- The library of string functions is similar in purpose to the C strings
- The cin statement only reads a string until it encounters white space. The use of the getline function reads until it finds a end of string terminator.

Inheritance 5%

  See http://www.ccpedagogy.com/cpp/class%20inheritance.pdf

Polymorphism 5%

- References to inherited data/functions is determined at compile time unless explicitly reference by a combination of the keyword virtual and the use of pointers.
- Use of same function name in two different classes in an inherited situation can cause problems.
- Best technique as an individual programmer is avoiding using same function name in multiple classes.
- Use this technique when working multiple programmers since probability of overlapping function names in likely.

  See http://www.ccpedagogy.com/cpp/Poly.pdf