**EE2310 C++程式設計 HW 6 (Chapter 9, 10) due: 5/30/2019**

**Part 1, Choice(s) 選擇題 (30%)**

1) The advantage of a linear search is that \_\_\_\_\_\_\_\_.

A) it is simple

B) it is efficient

C) it is fast

D) it can be used on unordered data

E) both A and D

2) A(n) \_\_\_\_\_\_\_\_ search is more efficient than a(n) \_\_\_\_\_\_\_\_ search.

A) string, double

B) integer, double

C) binary, linear

D) linear, binary

E) None of the above; all searches are equally efficient.

3) The linear search is adequate for searching through \_\_\_\_\_\_\_\_ arrays, but not through \_\_\_\_\_\_\_\_ ones.

A) int, double

B) char, string

C) ascending, descending

D) small, large

E) any regular, vector

4) A binary search begins by examining the \_\_\_\_\_\_\_\_ element of an array.

A) first

B) last

C) largest

D) middle

E) smallest

5) If the item being searched for is notin the array, binary search stops looking for it and reports that it is not there when \_\_\_\_\_\_\_\_.

A) array index first > array index last

B) Boolean variable found = false

C) Boolean variable found = true

D) it finds a value larger than the search key

E) it has examined all the elements in the array

6) A search can be performed on an array of \_\_\_\_\_\_\_\_ .

A) integers

B) strings

C) objects

D) All of the above

E) A and B, but not C

7) A sorting algorithm can be used to arrange a set of \_\_\_\_\_\_\_\_ in \_\_\_\_\_\_\_\_ order.

A) numeric values, ascending

B) numeric values, descending

C) strings, ascending

D) strings, descending

E) All of the above

8) The \_\_\_\_\_\_\_\_ sort usually performs fewer exchanges than the \_\_\_\_\_\_\_\_ sort.

A) bubble, selection

B) selection, bubble

C) binary, linear

D) linear, binary

E) linear, bubble

9) To find a value that is in an unordered array of 100 items, linear search must examine an **average** of \_\_\_\_\_\_\_\_ values.

A) 7

B) 10

C) 50

D) 100

E) 101

10) To determine that a item is not in an unordered array of 100 items, linear search must examine an **average** of \_\_\_\_\_\_\_\_ values.

A) 7

B) 10

C) 50

D) 100

E) 101

11) To locate a value in an ordered array of 100 items, binary search must examine **at most** \_\_\_\_\_\_\_\_ values.

A) 7

B) 10

C) 50

D) 100

E) 101

12) A bubble sort is being used to arrange the following set of numbers in ascending order:

7 5 3 9 2 6

After the first pass of the sort is completed, what order will the data be in?

A) 2 5 3 9 7 6

B) 5 7 3 9 2 6

C) 5 3 7 2 6 9

D) 2 3 5 6 7 9

E) None of the above

C

13) A selection sort is being used to arrange the following set of numbers in ascending order:

7 5 3 9 2 6

After the first pass of the sort is completed, what order will the data be in?

A) 2 5 3 9 7 6

B) 5 7 3 9 2 6

C) 5 3 7 2 6 9

D) 2 3 5 6 7 9

E) None of the above

14) When sorting an array of objects, if the values in the data member being sorted on are out of order for two objects, it is necessary to \_\_\_\_\_\_\_\_.

A) examine a different data member

B) swap these two data values

C) swap the two objects

D) swap one-by-one all data members in the two objects

E) stop the sort

C

15) We can measure the complexity of an algorithm that solves a computational problem by

determining the number of \_\_\_\_\_\_\_\_ for an input of size *n*.

A) output statements it has

B) times it loops

C) basic steps it requires

D) variables it uses

E) operations it performs

16) A \_\_\_\_\_\_\_\_ search uses a loop to sequentially step through an array.

A) binary

B) unary

C) linear

D) relative

E) bubble

17) The \_\_\_\_\_\_\_\_ search is adequate for searching through small arrays, but not through large ones.

A) binary

B) linear

C) blind

D) bubble

E) random

18) If a binary search is used to search for the number 4 in the 11-element array shown below, which value will the 4 be compared to first?

1 2 3 4 6 7 8 9 10 12 17

A) 1

B) 7

C) 8

D) 9

E) None of the above

19) When searching for a particular object in an array of objects, it is necessary to compare the \_\_\_\_\_\_\_\_ to the value in each examined object's \_\_\_\_\_\_\_\_ .

A) search key, private member data

B) key field, public member data

C) search key, public member data

D) search key, key field

E) key field, search key

D

20) To determine that a value is not present in an unordered array of 50 items, linear search must examine an **average** of \_\_\_\_\_\_\_\_ values.

A) 1

B) 6

C) 25

D) 50

E) 51

21) To locate a value that is in an ordered array of 50 items, linear search must examine **at most** \_\_\_\_\_\_\_\_ values.

A) 1

B) 6

C) 25

D) 50

E) 51

22) To locate a value in an ordered array of 50 items, binary search must examine **at most** \_\_\_\_\_\_\_\_ values.

A) 1

B) 6

C) 10

D) 25

E) 50

23) A \_\_\_\_\_\_\_\_ algorithm arranges data into some order.

A) sorting

B) searching

C) ordering

D) linear

E) binary

24) Sorted data can be ordered \_\_\_\_\_\_\_\_.

A) from lowest to highest value

B) from highest to lowest value

C) using a bubble sort algorithm

D) using a selection sort algorithm

E) in all of the above ways

25) When an array is sorted from highest to lowest, it is said to be in \_\_\_\_\_\_\_\_ order.

A) reverse

B) forward

C) descending

D) ascending

E) downward

26) A bubble sort is being used to arrange the following set of numbers in ascending order:

8 6 4 9 3 7

After the first pass of the sort is completed, what order will the data be in?

A) 3 4 6 7 8 9

B) 3 6 4 9 8 7

C) 6 4 8 3 7 9

D) 6 8 4 9 3 7

E) None of the above

C

27) A selection sort is being used to arrange the following set of numbers in ascending order:

8 6 4 9 3 7

After the first pass of the sort is completed, what order will the data be in?

A) 3 4 6 7 8 9

B) 3 6 4 9 8 7

C) 6 4 8 3 7 9

D) 6 8 4 9 3 7

E) None of the above

28) The \_\_\_\_\_\_\_\_ sort usually performs more exchanges than the \_\_\_\_\_\_\_\_ sort.

A) bubble, selection

B) selection, bubble

C) binary, linear

D) linear, binary

E) linear, bubble

29) Selection sort requires \_\_\_\_\_\_\_\_ passes to put **n** data items in order.

A) n

B) n/2

C) n/2 +1

D) n-1

E) n+1

30) We can estimate the \_\_\_\_\_\_\_\_ of an algorithm by counting the number of steps

it requires to solve a problem.

A) efficiency

B) number of lines of code

C) run time

D) code quality

E) result

31) The \_\_\_\_\_\_\_\_, also known as the address operator, returns the memory address of a variable.

A) asterisk ( \* )

B) ampersand ( & )

C) percent sign (%)

D) exclamation point ( ! )

E) None of the above

32) With pointer variables, you can \_\_\_\_\_\_\_\_ manipulate data stored in other variables.

A) never

B) seldom

C) indirectly

D) All of the above

E) None of the above

33) The code segment int \*ptr; has the same meaning as \_\_\_\_\_\_\_\_.

A) int ptr;

B) \*int ptr;

C) int ptr\*;

D) int\* ptr;

E) None of the above

34) When you work with a dereferenced pointer, you are actually working with \_\_\_\_\_\_\_\_.

A) a variable whose memory has been deallocated

B) a copy of the value pointed to by the pointer variable

C) the variable whose address is stored in the pointer variable

D) All of the above

E) None of the above

35) Which arithmetic operations can be performed on pointers?

A) All arithmetic operations that are legal in C++

B) Multiplication, division, addition, and subtraction

C) Addition , subtraction , preincrement, and postincrement

D) Only multiplication and addition

E) None of the above

36) A pointer may be initialized with \_\_\_\_\_\_\_\_.

A) the address of an existing object of the appropriate type

B) the value of a floating-point constant

C) the value of a floating-point variable

D) All of the above

E) None of the above

37) The statement double \*num; \_\_\_\_\_\_\_\_.

A) defines a variable of type double called num

B) defines and initializes a pointer variable called num

C) initializes a variable called \*num

D) defines a pointer variable called num

E) None of the above

38) When the less than ( < ) operator is used between two pointer variables, the expression is testing whether \_\_\_\_\_\_\_\_.

A) the value pointed to by the first is less than the value pointed to by the second

B) the value pointed to by the first is greater than the value pointed to by the second

C) the address of the first variable comes before the address of the second variable in the computer's memory

D) the first variable was declared before the second variable

E) None of the above

39) Assuming that arr is an array identifier, the statement sum += \*arr; \_\_\_\_\_\_\_\_.

A) is illegal in C++

B) will always result in a compiler error

C) adds the value stored in arr[0] to sum

D) adds the address of the pointer arr to sum

E) None of the above

40) The delete operator should only be used on pointers that \_\_\_\_\_\_\_\_.

A) have not yet been used

B) have been correctly initialized

C) point to storage allocated by the new operator

D) are appropriately dereferenced

E) None of the above

41) A function may return a pointer, but the programmer must ensure that the pointer \_\_\_\_\_\_\_\_.

A) is pointing to an object that is still valid after the return of the function

B) has been assigned an address

C) was received as a parameter by the function

D) has not previously been returned by another function

E) None of the above

42) Which of the following statements is not valid C++ code?

A) int ptr = &num1;

B) int ptr = int \*num1;

C) float num1 = &ptr2;

D) All of the above are valid

E) All of the above are invalid

43) Which of the following statements correctly deletes a dynamically-allocated array pointed to by p?

A) delete p;

B) p delete[ ];

C) delete [ ] p;

D) delete array p;

E) None of the above

44) You may use a pointer to a structure as a \_\_\_\_\_\_\_\_.

A) function parameter

B) structure member

C) function return type

D) All of the above

E) None of the above

45) If arr is an array identifier and k is an integer, the expression arr[k] is equivalent to

A) \*(arr + k)

B) \*arr + k

C) &arr[k]

D) arr + k

E) None of the above

46) To dereference a structure pointer and simultaneously access a member of the structure, the appropriate operator to use is \_\_\_\_\_\_\_\_.

A) the ampersand, &

B) an asterisk, \*

C) the structure pointer operator, ->

D) the dereference operator, <-

E) None of the above

47) When the \_\_\_\_\_\_\_\_ is placed in front of a variable name, it returns the address of that variable.

A) asterisk ( \* )

B) conditional operator

C) ampersand ( & )

D) semicolon ( ; )

E) None of the above

48) The statement cout << &num1; will output \_\_\_\_\_\_\_\_.

A) the value stored in the variable called num1

B) the memory address of the variable called num1

C) the number 1

D) the string "&num1"

E) None of the above

49) A pointer variable is designed to store \_\_\_\_\_\_\_\_.

A) any legal C++ value

B) only floating-point values

C) a memory address

D) an integer

E) None of the above

50) The statement int \*ptr; means \_\_\_\_\_\_\_\_.

A) the variable called ptr will store an integer value

B) the variable called \*ptr will store an asterisk and an integer value

C) ptr is a pointer variable that will store the address of an integer variable

D) All of the above

E) None of the above

51) The statement cout << \*ptr; will output \_\_\_\_\_\_\_\_.

A) the value stored in the variable whose address is contained in ptr

B) the string "\*ptr"

C) the address of the variable stored in ptr

D) the address of the variable whose address is stored in ptr

E) None of the above

52) The \_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_ operators can respectively be used to increment and decrement a pointer variable.

A) dereferencing, indirection

B) modulus, division

C) ++, --

D) All of the above

E) None of the above

53) The statement cin >> \*p; \_\_\_\_\_\_\_\_.

A) stores the keyboard input into the variable p

B) stores the keyboard input into the pointer called p

C) is illegal in C++

D) stores the keyboard input into the variable pointed to by p

E) None of the above

54) Dynamic memory allocation occurs \_\_\_\_\_\_\_\_.

A) when a variable is created by the compiler

B) when a variable is created at run-time

C) when a pointer fails to dereference the right variable

D) when a pointer is assigned an incorrect address

E) None of the above

55) The statement int \*ptr = new int; acquires memory to hold an integer and then \_\_\_\_\_\_\_\_.

A) initializes the allocated memory to 0

B) assigns an integer value to the variable called ptr

C) sets ptr to point to the allocated memory

D) creates a new pointer called int

E) None of the above

56) Any time you use the new operator, it is good practice to \_\_\_\_\_\_\_\_.

A) use delete afterwards to free the memory allocated by new

B) use a preprocessor directive

C) clear the data from the old operator

D) All of the above

E) None of the above

57) If dynamically allocated memory is not freed, \_\_\_\_\_\_\_\_.

A) the system may run out of memory

B) it results in a compiler error

C) a run-time error informs your user that the program did not free memory space

D) the source code will not link correctly

E) None of the above

58) A reason for passing a pointer to a function is \_\_\_\_\_\_\_\_.

A) to avoid the overhead of copying large data structures

B) to allow the called function to modify a variable accessible to the calling function

C) to allow easy access to data in the function that is being called

D) A and B are both true

E) None of the above

59) A pointer variable may be initialized with \_\_\_\_\_\_\_\_.

A) any non-zero integer value

B) the address of an existing variable of the appropriate type

C) A and B are both true

D) None of the above

60) If a variable occupies more than one byte of memory, its address is \_\_\_\_\_\_\_\_.

A) the address of the last byte of storage allocated to it

B) the average of the addresses used to store the variable

C) the address of the first byte of storage allocated to it

D) general delivery

E) None of the above

**Part 2, True/False 是非題 (10%)**

1. Using a linear search, you are more likely to find an item than if you use a binary search.

F

1. When searching for an item in an **unordered** set of data, binary search can find the item more quickly than linear search.

F

3) When sorting an array of objects or structures, one must decide which data item to sort on.

T

4) Bubble sort and selection sort can also be used with STL vectors.

T

5) If algorithm A requires 2*n* + 1basic operations to process an input of size *n*, and Algorithm B requires 3*n + 2* basic operations to process the same input, algorithm A is considered to be more efficient than Algorithm B.

T

6) If algorithm A requires 2*n* + 1basic operations to process an input of size *n*, and Algorithm B requires 3*n + 2* basic operations to process the same input, algorithms A and B are considered to be equally efficient.

F

7) A binary search requires that the elements be in order.

T

8) Using a binary search, you are more likely to find an item than if you use a linear search.

F

9) Any sorting algorithm, such as bubble sort or selection sort, that can be used on data stored in an array can also be used on data stored in a vector.

T

10) When sorting an array of objects, if the values in the data member being sorted on are out of order for two objects, those two data values should be swapped.

F

1. The statement Rectangle \* boxPtr; defines a variable boxPtr to be a pointer pointing to a type Rectangle.

T

1. With pointer variables you can access, but you cannot modify, data in other variables.

F

1. An array name is a pointer constant because the address it represents cannot be changed during run-time.

T

1. Variables cannot be created when a program is already running.

F

1. The expression s->m has the same meaning as (\*s).m

T

1. A pointer can be passed as an argument to a function.

T

1. Any arithmetic operation may be performed on pointers.

F

1. The ampersand (&) is used to dereference a pointer variable in C++.

T

1. The expression \*s->p; is only meaningful if s is a pointer to a structure and p is a pointer that is a member of that structure.

T

1. It is possible for a structure to contain as a member a pointer to its own structure type.

T

**Part 3, Coding 程式題(60%)**

1. **Modified Bin Manager Class**

Modify the BinManager class you wrote for Programming #6 in HW5 to overload its getQuantity, addParts, and removeParts functions as shown here:

bool addParts(string itemDescription, int q);

bool removeParts(string itemDescription, int q)

int getQuantity(string itemDescription);

These new functions allow parts to be added, parts to be removed, and the quantity in stock for a particular item to be retrieved by using an **item description**, rather than a bin number, as an argument. In addition to writing the three overloaded functions, you will need to create a private BinManager class function that uses the item description as a search key to locate the index of the desired bin.

Test the new class functions with the same client program you wrote for Programming #6 in HW5, modifying it to call the new functions. Be sure to use some descriptions that match bins in the array and some that do not.

As you did in the previous Bin Manager program, if an add or remove operation is successfully carried out, make the function return true. If it cannot be done—for example, because the string passed to it does not match any item description in the array—make the function return false. If the getQuantity function cannot locate any item whose description matches the one passed to it, make it return −1.

1. **Statistics Report of Monthly Rainfall**

Based on programming #4 in HW5, redesign your program so that (a) it uses **dynamic storage instead of fixed size array to store rainfall data**, (b) in addition to the original function and report, it also displays the name of each month in the period (at least a year) and its rainfall amount, sorted in order of rainfall from highest to lowest. You should modify your program accordingly. Make the program modular by calling on different functions to input the rainfall amounts, to sort the data, and to display the data.

1. **Sorting Benchmarks**

Write a program that uses two identical arrays of at least 20 integers generated randomly. It should call a function that uses the **bubble sort** algorithm to sort one of the arrays in ascending order. The function should count the number of exchanges it makes. The program should then call a function that uses the **selection sort** algorithm to sort the other array. It should also count the number of exchanges it makes. Repeat the process 20 times and display these values of each time as well as the average values of exchanges for the two sorting methods on the screen.

1. **Search Benchmarks**

Write a program that has an array of at least 20 integers generated randomly. It should call a function that uses the **linear search algorithm** to locate one of the values. The function should keep a count of the number of comparisons it makes until it finds the value. The program then should call a function that uses the **binary search algorithm** to locate the same value. It should also keep count of the number of comparisons it makes. Display these values on the screen.