Study guide for Test #1 in CS 300

Review the questions numbered 1 to 30 found at the end of Chapter 10 on pages 689 to 692.

Review the questions numbered 1 to 7 found at the end of Chapter 14 on pages 928 to 930.

Review the questions numbered 1 to 17 found at the end of Chapter 17 on pages 1064 to

Chapter 10 answers - <https://quizlet.com/281152757/flashcards> , <https://quizlet.com/45746905/flashcards>

1. Each byte in memory is assigned a unique \_\_\_\_\_\_\_\_\_\_ - address

2. The \_\_\_\_\_\_\_\_\_\_ operator can be used to determine a variable’s address - ampersand (&)

3. \_\_\_\_\_\_\_\_\_\_ variables are designed to hold addresses - pointer

4. The \_\_\_\_\_\_\_\_\_\_ operator can be used to work with the variable a pointer points to - Indirection (\*)

5. Array names can be used as \_\_\_\_\_\_\_\_\_\_ and vice versa - pointers

6. Creating variables while a program is running is called \_\_\_\_\_\_\_\_\_\_ - Dynamic memory allocation.

7. The \_\_\_\_\_\_\_\_\_\_ operator is used to dynamically allocate memory - new

8. If the new operator cannot allocate the amount of memory requested, it throws \_\_\_\_\_\_\_\_\_\_ - null

9. A pointer that contains the address 0 is called a(n) \_\_\_\_\_\_\_\_\_\_ pointer - null

10. When a program is finished with a chunk of dynamically allocated memory, it should free it with the \_\_\_\_\_\_\_\_\_\_ operator - delete

11. You should only use the delete operator to deallocate memory that was dynamically acquired with the \_\_\_\_\_\_\_\_\_\_ operator - new

12. What does the indirection operator do? Dereferences the pointer; allows you to work with the value the pointer is pointing to.

13. int x = 7; int \*ptr = &x; What will be displayed if you send the expression \*iptr to cout? What happens if you send the expression ptr to cout? ---- Sending \*iptr to cout will display 7. Sending ptr to cout will display the address of x.

14. Name two different uses for the C++ operator \* - \* is used as the multiplication operator, to define a pointer variable (int\*ptr) and as the indirection operator(\*ptr=100)

15. Which arithmetic operations can be applied to pointers? - addition, subtraction (++) (--)

16. Assuming that ptr is a pointer to an int, what happens when you add 4 to it? - It multiplies 4 by the number of bytes for an int then adds it to the memory address. The value 16 is added to the memory address of ptr.

17. int numbers [] = {2, 4, 6, 8, 10};

What will the following statement display?

cout << \*(numbers + 3) << endl; - 8

18. What is the purpose of the new operator? new is Used for Dynamic Memory Allocation to set aside a chunk of memory for a specific data type.

19. What happens when a program uses the new operator to allocate a block of memory, but the amount of requested memory isn’t available? How do programs written with older compilers handle this? It throws an exception and terminates the program. In older compilers it returns 0.

20. Under what circumstances can you successfully return a pointer from a function? You can successfully return a pointer from a function When a pointer variable is passed into a function as a parameter and when A pointer is indicated to dynamically allocated memory

21. What is the purpose of the delete operator? To free memory that was created by the new operator.

22. What is the difference between a pointer to a constant and a constant pointer? Pointer to a constant- a pointer points to a constant value, it cannot change any values inside the constant.

constant pointer- once the pointer is initialized with an address, it cannot point to anything else.

23. Show C++ code for defining a variable ptr that is a pointer to a constant int. - const int \*ptr;

24. Show C++ code for defining a variable ptr that is a constant pointer to int.- int \* const ptr;

25. Consider the function

void change(int \*p)

{

\*p = 20;

} Show how to call the change function so that it sets the integer variable

int i;

to 20. - change(&i);

26. void modify(int & x)

{

x = 10;

}

Show how to call the modify function so that it sets the integer

int i;

to 10. - modify(i);

27. Write a function whose prototype is

void exchange(int \*p, int \*q);

that takes two pointers to integer variables and exchanges the values in those variables –

void exchange(int p, int q)

{

int temp = \*p;

p = q;

\*q = temp;

}

28. Write a function

void switchEnds(int \*array, int size); that is passed the address of the beginning of an array and the size of the array. The function swaps the values in the first and last entries of the array .

void switchEnds(int \*array, int size){

int temp = \*(array);

(array) = (array+size-1);

\*(array+size-1) = temp;

}

29. int a[5] = {0, 10, 20, 30, 40};

int k = 3;

int \*p = a + 1;

determine the output from each of the following statements:

A) cout << a[k]; - 30

B) cout << \*(a + k); - 30

C) cout << \*a; - 0

D) cout << a[\*a]; - 0

E) cout << a[\*a + 2]; - 20

F) cout << \*p; - 10

G) cout << p[0]; - 10

H) cout << p[1]; - 20

30. Each of the following declarations and program segments has errors. Locate as many

as you can.

A) int ptr\*; star in the wrong place

B) int x, \*ptr;

&x = ptr; change around assignors

C) int x, \*ptr;

\*ptr = &x; take out star

D) int x, \*ptr;

ptr = &x;

ptr = 100; // Store 100 in x

cout << x << endl; star infront of ptr for 100

E) int numbers[] = {10, 20, 30, 40, 50};

cout << "The third element in the array is ";

cout << \*(numbers + 3 ) << endl;

F) int values[20], \*iptr;

iptr = values;

iptr \*= 2; take out star

G) double level;

int dPtr = &level; change int to double and put star infront of dptr

H) int \*iptr = &ivalue;

int ivalue; switch statements to put 2nd one first

I) void doubleVal(int \* val)

{

\*val \*= 2;

}

J) int \*pint;

new pint; pint = new int

K) int \*pint;

pint = new int;

pint = 100;

L) int \*pint;

pint = new int[100]; // Allocate memory

.

.

Process the array

.

.

delete [] pint;// Free memory

M) int \*getNum()

{

int wholeNum;

cout << "Enter a number: ";

cin >> wholeNum;

return &wholeNum;

}

1. What type of recursive function do you think would be more difficult to debug; one that uses direct recursion, or one that uses indirect recursion? Why? indirect recursion because it jumps to another function

2. Which repetition approach is less efficient; a loop or a recursive function? Why? recursive functions are less efficient because of the overhead involved with calling a function in C++. Memory must be allocated and address much be stored.

recursive functions are usually shorter, cleaner, and slower.

3. When should you choose a recursive algorithm over an iterative algorithm? In situations where the recursive algorithm is easier to design. Specifically, situations where a problem can be broken down into small repetitions of very similar problems.

4. The \_\_\_\_\_\_\_\_\_\_ of recursion is the number of times a function calls itself. depth

5. \_\_\_\_\_\_\_\_\_\_ recursion is when a function explicitly calls itself. direct

6. \_\_\_\_\_\_\_\_\_\_ recursion is when function A calls function B, which in turn calls function A. indirect

7. . What is the output of the following programs?

A) #include <iostream>

using namespace std;

int function(int);

int main()

{

int x = 10;

cout << function(x) << endl;

return 0;

}

int function(int num)

{

if (num <= 0)

return 0;

else

return function(num - 1) + num;

} 55

1. The \_\_\_\_\_\_\_\_\_\_ points to the first node in a linked list. head

2. A data structure that points to an object of the same type as itself is known as a(n)

\_\_\_\_\_\_\_\_\_\_ data structure. Self-referential

3. To indicate that a linked list is empty, you should set the pointer to its head to the

value \_\_\_\_\_\_\_\_\_\_. Null.

4. \_\_\_\_\_\_\_\_\_\_ a node means adding it to the end of a list. appending

5. \_\_\_\_\_\_\_\_\_\_ a node means adding it to a list, but not necessarily to the end. inserting

6. \_\_\_\_\_\_\_\_\_\_ a list means traveling through the list. traversing

7. In a(n) \_\_\_\_\_\_\_\_\_\_ list, the last node has a pointer to the first node. circularly

8. In a(n) \_\_\_\_\_\_\_\_\_\_ list, each node has a pointer to the one before it and the one after it. doubly

9. Using the ListNode structure introduced in this chapter, write a function

void printFirst(ListNode \*ptr)

that prints the value stored in the first node of a list passed to it as parameter. The function

should print an error message and terminate the program if the list passed to it is empty.

10. Write a function

void printSecond(ListNode \*ptr)

that prints the value stored in the second node of a list passed to it as parameter.

The function should print an error message and terminate the program if the list

passed to it has less than two nodes.

11. Write a function

double lastValue(ListNode \*ptr)

that returns the value stored in the last node of a nonempty list passed to it as

parameter. The function should print an error message and terminate the program if

the list passed to it is empty.

92 // Erase the current image

93 sequence[k]->eraseAt(pos, 3);

94 // Move to next image in the rotation and next position

95 k = (k+1) % 3;

96 pos = pos + 8;

97 }

98 sequence[k]->displayAt(pos, 3);

99 return 0;

100 }

12. Write a function

ListNode \*removeFirst(ListNode \*ptr)

that is passed a linked list as parameter, and returns the tail of the list: that is, it

removes the first node and returns what is left. The function should deallocate the

storage of the removed node. The function returns NULL if the list passed to it is

empty.

13. Write a function

ListNode \*ListConcat(ListNode \*list1, ListNode \*list2)

That concatenates the items in list2 to the end of list1 and returns the resulting

list.

Predict the Output

For each of the following program fragments, predict what the output will be.

14. ListNode \*p = new ListNode(56.4);

p = new ListNode(34.2, p);

cout << (\*p).value << endl << p->value;

15. ListNode \*p = new ListNode(56.4);

p = new ListNode(34.2, p);

ListNode \*q = p->next;

cout << q->value;

16. ListNode \*p = new ListNode(56.4, new ListNode(31.5));

ListNode \*q = p;

while (q->next->next != NULL)

q = q->next;

cout << q->value;

Find the Errors

17. Each of the following member functions for performing an operation on a linked list

of type NumberList has at least one error. Explain what is wrong and how to fix it.

A) NumberList::printList( )

{

while(head)

{

cout << head->value;

head = head->next;

}

}

B) NumberList::printList( )

{

ListNode \*p = head;

while (p->next)

{

cout << p->value;

p = p->next;

}

}

1066 Chapter 17 Linked Lists

C) NumberList::printList( )

{

ListNode \*p = head;

while(p)

{

cout << p->value;

p++;

}

}

D) NumberList::~NumberList()

{

ListNode \*nodePtr, \*nextNode;

nodePtr = head;

while (nodePtr != NULL)

{

nextNode = nodePtr->next;

nodePtr->next = NULL;

nodePtr = nextNode;

}

}