2. The & operator can be used to determine a variable's address.

4. The indirection operator can be used to work with the variable a pointer points to.

6. Creating variables while a program is running is called Dynamic Memory Allocation.

8. If the new operator cannot allocate the amount of memory requested,it throws bad alloc.

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

12. What does the indirection operator do?

The indirection operator which is called by the \* symbol allows you to obtain the value that the pointer is pointing to.This is most commonly referred to as dereferencing the variable which is taking the value of where the memory address is.

14. Name two different uses for the c++ operator \*.

It is used to multiply two integers such as 2 \* 2 would be 4 and also is used to reference the indirection operator when in context to pointers such as in the previous question.

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 it adds it to the memory address.Simply put the pointer would advance 4 places in the memory.

18.What is the purpose of the new operator?

It tells the compiler to set aside a certain chunk of memory to store a variable of that type.memory should always be freed up whenever it is not needed anymore when you use the new operator by using the delete operator.

20. Under what circumstances can you successfully return a pointer from a function?

Circumstances that a pointer can be returned from a function are when a pointer to an object was passed into the function as an argument and when a pointer is set to a dynamically allocated object/chunk of memory.

22. What is the difference between a pointer to a constant and a constant pointer?

A pointer to a constant which is when a pointer points to a value that is constant and such cannot change any values inside the constant.Constant pointer which is when a pointer is initialized to a certain memory address it cannot point to anything else.Each of these has a specific purpose.

24. Show C++ code for defining a variable ptr that is a constant pointer to int.

int \* const ptr;

//spacing does not matter

26. Name the header file that needs to be included in a program that uses smart pointers.

<memory>

28. What does the get () method of the unique\_ptr class do?

It is used when a pointer is passed to a function but does not know how to use smart pointers.

30. List three different operations that are permitted on raw pointers but not on unique\_ptr objects.

++, +=, uptr3 = uptr 1

32. What problems are likely to occur if you have the following declaration in your program?

shared\_ptr<double []> sDPtr;

You would get an error

//The double array object elements would be destroyed

34. Consider the function

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);

//this passes i to the function then assigns 10

36. 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)

{

swap(array,array+size-1);

//call the function wich will swap the values

}

void swap(int \*p, int \*q)

{

// store in temp then swap values

int temp = \*p;

\*p = \*q;

\*q = temp;

}

38. Each of the following declarations and program segments has errors. Locate as many as you can.

A) int ptr\*;

The indirection operator (\*) should be before ptr.

int\* ptr;

B) int x, \*ptr;

&x=ptr;

Should be

Int x,\*ptr;

ptr=&x;

C) int x, \*ptr; \*ptr = &x;

Should be

Int x,\*ptr;

ptr=&x;

//dont use the inderection operator twice

D) int x, \*ptr; ptr = &x; ptr = 100; II Store 100 in x cout << x << endl;

Should be

Int x,\*ptr;ptr=&x;

\*ptr=100;

cout<<x<<endl;

E) int numbers[] = {10, 20, 30, 40, 50}; cout << ''The third element in the array is ''; cout << \*numbers + 3 << endl;

Should be

Int numbers[]={10,20,30,40,50};

cout<<”The third element in the array is “<<\*(numbers+3)<<endl;

F) int values[20], \*iptr; iptr = values; iptr \*= 2;

Should be

Int values[20], int \*iptr;

iptr=values;

//will be the first element in the array

\*iptr=2;

G) double level; int dPtr = &level;

Should be

Double level;

Double\*dPtr;

//should be a pointer to access the address.

dPtr=&level;

H) int \*iptr = &ivalue; int ivalue;

Should be

Int \*iptr;

Int ivalue;

iptr=&ivalue;

I) void doubleVal(int val) { \*val \*= 2; }

Should be

Void double Val(int\* val)

{

\*val \*=2

}

J) int \*pint; new pint;

Should be

//be declared all together to initilizae space

Int \*pint=new int;

K) int \*pint; pint = new int; pint = 100;

Should be

Int \*pint;

pint= newint

\*pint=100;

L) int \*pint; pint = new int[lOO]; //Allocate memory • • Process the array • • delete pint;// Free memory

//there is an array so you cant say delete pint because pint is refering to the array

Should be

Delete pint[];

M) int \*getNum( )

{ int wholeNum; cout << ''Enter a number: ''; cin >> wholeNum; return & wholeNum;

}

Should be

Int getNum;

cout<<”Enter a number”;cin>>wholeNum;return wholeNum;} //should return the value of the variable