CS 150 Midterm 2

Share

9 studiers today ★ 4.0 (2 reviews)

Terms in this set (20)

Examine the code below and match the statements following it.
`int mystery1(int n, int a, int b) {`
`if (n == 0) return a;`
`if (n == 1) return b;`
`return mystery1(n - 1, b, a + b);`
`}`
`int mystery2(int n) {`
`return mystery1(n, 0, 1);`
`}`
	mystery2 is a recursive wrapper -> True
	mystery2 completes for all possible inputs -> False
	if (n == 0) is a recursive case -> False
	These functions illustrate how inefficient recursion is. -> False

Examine the code below:
`int mystery1(int n, int a, int b) {`
`if (n == 0) return a;`
`if (n == 1) return b;`
`return mystery1(n - 1, b, a + b);`
`}`
`int mystery2(int n) {`
`return mystery1(n, 0, 1);`
`}`
	- mystery2 is a recursive wrapper
	- if (n == 0) is a base case

Examine the code below and match the statements following it.
`int mystery3(int n) {`
`if (n < 2) return 1;`
`return n * mystery3(n - 1);`
`}`
	The algorithm implemented is: -> Factorial
	mystery3 has a stack overflow for some numbers. -> False
	mystery3 completes for all inputs -> True
	mystery3 correctly implements its algorithm -> True
	mystery3 is a recursive wrapper -> False
	if (n < 2) is a . . . -> base case
	mystery3 returns the correct answer for all inputs -> False
	mystery3 is efficient -> True

Examine the code below:
 	
int mystery3(int n) {
if (n < 2) return 1;
return n * mystery3(n - 1);
}
	- mystery3 is efficient
	- In mysterm3, if (n < 2) is a base case

Examine this code. Which is the best prototype?
 	
string s = "dog";
upper(s);
cout << s << endl; // DOG
	string upper(string&)

An undeclared error message is a runtime error.	False
The statement x = cin.get(ch) returns the next character from input and stores it in x.	False

This idiomatic pattern is used to count from one value to another.
 	
for (int i = 0; i < 10; i++)
cout << i;
cout << endl;
	False

Header files may contain the statement using namespace std;	False

Match the following code the the answers below.
 	
template <typename T, typename U>
U pickle(T& a, const U& b) {
a += b;
return b;
}
 	
int main()
{
int x = 42;
auto a = pickle(x, 4.5);
cout << a << endl;
cout << x << endl;
}
	Inside main, the variable a is type: -> double
	
	Inside main, the value printed for a is: -> 4.5
	
	Inside main, the value printed for x is: -> 46

Using the loop-building strategy from the lessons, which of these are part of the loop mechanics?	- loop bounds - advancing the loop - bounds precondition
[1] File containing the declarations or prototypes [2] Program which uses the functions in a library. [3] File containing the function definitions [4] File which contains instructions for building your program	[1] interface [2] client [3] implementation [4] makefile
What changes about this function if lines 4 and 5 are swapped? 1. void myfun(const string& word) 2. { 3. if (word.size() == 0) { return; } 4. myfun(word.substr(1)); 5. cout << word[0]; 6. }	reverses the order in which the characters of the string are printed

What Java and other OO languages call a superclass, C++ calls a _______.

base class

Examine the following variables and function calls Match each item with the correct statement below.

```
int able = 3;
int baker = f1(able);
cout << able << baker << endl; // 64

string charlie;
f2("hello", charlie);
cout << charlie << endl; // Hello Carl
```

Returned value -> baker

Output argument (parameter) -> charlie

Input argument (parameter) -> hello

Input/output argument (parameter) -> able

What prints here?

```
double a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
}
cout << endl;
```

Does not compile

If an output stream's file is missing when you try to open it, its fail() member function returns true.

False

What does this code do?

```
ifstream in("temp.txt");
char x;
int i{0};
while (in >> x) i++;
cout << i << endl;
```

NOT - Counts the number of words in the file

How many lines of output are printed?

```
int i = 0;
int j = 0;
while (i < 25)
{
i = i + 2; j++;
}
cout << j << endl;
```

NOT - 12

This loop:

```
char c;
while (c = in.get())
{
cout << c << endl;
}
```

NOT - illustrates line-based stream processing