This is not an assignment, just an exercise

Coverage. Consider the following program fragment (a simple version of Kadane's algorithm for the maximums sub-array problem) and test suite:

```
bool subarr(int i, int j, int k)
  int maxsum = i;
  int lastsum = i;
  if (lastsum < 0)
    lastsum = j;
  else
    lastsum += j;
  if (lastsum > maxsum)
    maxsum = lastsum;
  if (lastsum < 0)
    lastsum = k;
  else
    lastsum += k;
  if (lastsum > maxsum)
    maxsum = lastsum;
  return maxsum;
}
```

Inputs			Output
i	j	k	result
-3	-1	2	2
3	-1	2	4

Control-Flow-Based Coverage Criteria. Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (assume that the term "decision" refers to all Boolean expressions in the program).

	satisfied	
Criterion	yes	no
path coverage		
statement coverage		
branch coverage		
decision coverage		
condition/decision coverage		

Data-Flow-Based Coverage Criteria. Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (here, the parameters of the function do not constitute definitions):

	satisfied	
Criterion	yes	no
all-defs		
all-c-uses		
all-p-uses		
all-c-uses/some-p-uses		
all-p-uses/some-c-uses		
all-uses		
all-du-paths		

If the test-suite from above does not satisfy the coverage criteria listed below, augment it with test-cases such that these criteria are satisfied. If full coverage cannot be achieved for one or more of these criteria, explain why.

decision coverage

Inputs			Output
`i`	`j`	`k`	result

all-p-uses/some-c-uses

Inputs			Output
`i`	`j`	`k`	result

MC/DC

Inputs			Output
`i`	`j`	`k`	result

Provide sufficiently many test-cases to guarantee modified condition/decision coverage for the following program fragment:

```
bool foo(int x, int y) {
  return ((x < y) || (y >> 2 == 0 ));
}
```

Input		Output
`x`	`y`	result