

The following fragment of code shows a binary search routine.

```
int binsearch(int X, int V[], int n){
    int low, high, mid;
    low = 0;
    high = n - 1;
    while (low <= high) {
        mid = (low + high)/2;
        if (X < V[mid])
            high = mid - 1;
        else if (X > V[mid])
            low = mid + 1;
        else
            return mid;
    }
    return -1;
}
```

Exercises:

You are given the binary search routine above. The input array V is assumed to be sorted in ascending order, n is the array size, and you want to find the index of an element X in the array. If X is not found in the array, the routine is supposed to return -1 .

The first eight questions refer to the `binsearch()` function.

1. Draw a CFG for `binsearch()`.
2. From the CFG, identify a set of entry–exit paths to satisfy the complete statement coverage criterion.
3. Identify additional paths, if necessary, to satisfy the complete branch coverage criterion.
4. For each path identified above, derive their path predicate expressions.
5. Solve the path predicate expressions to generate test input and compute the corresponding expected outcomes.
6. Are all the selected paths feasible? If not, select and show that a path is infeasible, if it exists.
7. Can you introduce two faults in the routine so that these go undetected by your test cases designed for complete branch coverage?
8. Suggest a general way to detect the kinds of faults introduced in the previous step.
9. What are the limitations of control flow–based testing?
10. Show that branch coverage includes statement coverage.