Computer Science I Lab (CSC 2111)

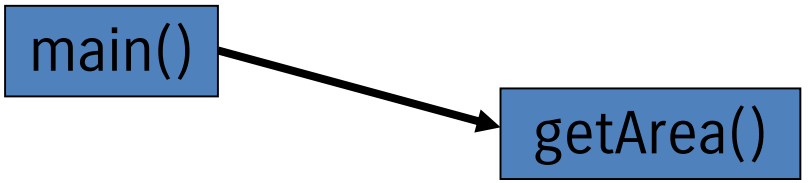
Lab 20

Objectives

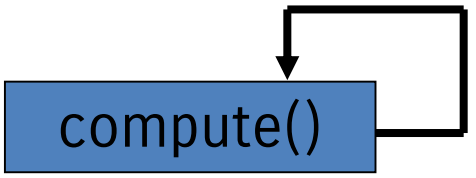
1. Understand basic recursion

Concept

- **Regular Method:** A method that call other functions. For example, the main() method calls the getArea() function.



- **Recursive Method:** A recursive method is a method that calls itself.



Recursive Method

The recurrsive case

- Divide the given problem into smaller parts.
- Invokes itself to compute the smaller parts, which **eventually reaches the base case**

A base (or stopping) case

- Provides a direct (non-recursive) solution for the base case
- Code first tests for the base case. If the base case is not reached yet, it goes to the recursive case

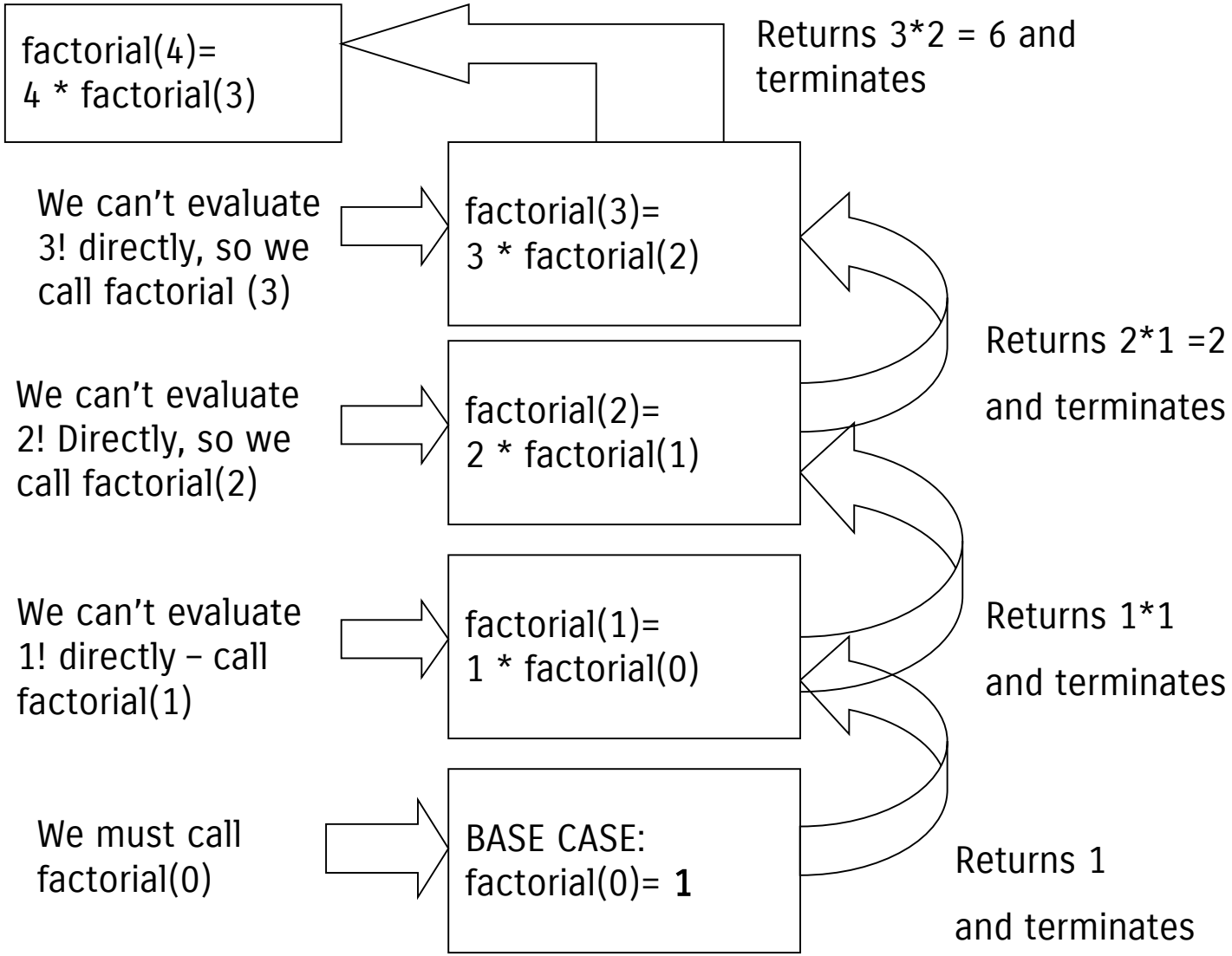
Example 1

Computing factorial (x!):

```
int factorial (int x)
{
        //base case
        if (x ==0) return 1;

        //recurrence case
        return x * factorial (x – 1);
}
```

Trace of a call to Factorial: int z = factorial(4)



finally, factorial(4) computes 4*6, returns 24, and terminates

Example 2

Implement a recursive function that returns true if the string parameter is a palindrome, false otherwise.

Example 2 (i)

```
bool palindromic(string m, int low, int high) {
      if (low >= high) {
            /* Base Case */
            return true;
      }
      if (m[low] != m[high]) {
            /* Shortcut */
            return false;
      }
      /* Recursive Step */
      return true && palindromic(m, low + 1, high - 1);
}
```

Example 2 (ii)

```
int main(void) {
	char p[] = "racecar";
	char np[] = "houseboat";
	if (palindromic(p, 0, strlen(p) -1 )) {
		cout << p << " is a palindrome" << endl;
	}
	else {
		cout << p << " is not a palindrome" << endl;
	}
	if (palindromic(np, 0, strlen(np) - 1)) {
		cout << np << " is a palindrome" << endl;
	}
	else {
		cout << np << " is not a palindrome" << endl;
	}
	return 0;
}
```

Output

```
racecar is a palindrome
houseboat is not a palindrome
```