|  |  |
| --- | --- |
| **Data Structures & Algorithms**  Diploma in IT, CSF  Year 2 (2024/25) Semester 4 | **Week 7** |
| **1 Hour** |
| **Tutorial 7 – Recursion** | |

|  |
| --- |
| **IMPORTANT**   * Upload all your answers to Brightspace by the designated time stated in Brightspace. |

1. Write a recursive function that will compute the sum of the first n integers in an array of at least n integers.

Hint: begin with the nth integer.

#include <iostream>

#include <string>

#include <vector>

using namespace std;

int sumFirstN(const vector<int>& arr, int n) {

    // Base case: if n is 0, return 0

    if (n == 0) {

        return 0;

    }

    // Recursive case: return the sum of the first n-1 elements plus the nth element

    return arr[n - 1] + sumFirstN(arr, n - 1);

}

int main() {

    vector<int> arr = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    int n = 5;

    int result = sumFirstN(arr, n);

    cout << "Sum of the first " << n << " integers: " << result << endl;

    return 0;

}

2. Describe the problem with the following recursive function:

// void functions don’t need to return a value but can write return;

void printNum(int n)

{

cout << n <<endl;

printNum(n-1);

}

1. No base case to stop the recursion, function will keep calling itself infinitely, causing stack overflow
2. N-1 is called indefinitely

3. Given an integer n > 0, write a recursive function that returns the sum of 1 through n.

#include <iostream>

#include <string>

#include <vector>

using namespace std;

int sumToN(int n) {

    // Base case: if n is 0, return 0

    if (n == 1) {

        return 1;

    }

    // Recursive case: return the sum of the first n-1 elements plus the nth element

    return n + sumToN(n - 1);

}

int main() {

    int n = 5;

    std::cout << "Sum of numbers from 1 to " << n << " is: " << sumToN(n) << std::endl;

    return 0;

}

4. Consider the following program:

1. int f(int n);
2. int main()
3. {
4. cout<< "The value of f(8) is " << f(8) << endl;
5. return 0;
6. }
7. // pre: n >= 0
8. int f(int n)
9. {
10. cout << "Function entered with n = " << n << endl;
11. if ( (n >= 0) && (n <= 2) )
12. return n + 1;
13. else
14. return f(n-2) \* f(n-4);
15. }

Show the exact output of the program. What argument values, if any, could you pass to the function f to cause an infinite recursion?

Function entered with n = 8

Function entered with n = 6

Function entered with n = 4

Function entered with n = 2

Function entered with n = 0

Function entered with n = 2

Function entered with n = 4

The value of f(8) is 27

When N becomes a negative value, it will cause an infinite recursion. Some examples is f(3) and f(-2)