|  |  |
| --- | --- |
| **Data Structures & Algorithms**  Diploma in CSF, IT  Year 2/3 (2020/21) Semester 4/6 | **Week 7** |
| **1 Hour** |
| **Tutorial 7 – Recursion** | |

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.

void printsum(int array[] ,int n , int value){

if ( n == 0){

value+=array[n];

cout << value <<endl;

} else {

Value+=array[n];

printsum(array,n,value);

}

}

int sum(int arr[], int n){

if (n==0)//

return 0;

else

return array[n-1]+ sum(array,n-1);

}

2. Describe the problem with the following recursive function:

void printNum(int n)

{

cout << n <<endl;

printNum(n-1);

}

The recursive function did not provide a base case.Therefore creating a infinite recursion call

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

Int recursiveint(int n){

If (n > 0){

Return n;

} else {

recursiveint(n+1);

}

}

int sum(int n){

if (n==1)

return 1;

else

return n+sum(n-1);

}

4. Consider the following program:

int f(int n);

int main()

{

cout<< "The value of f(8) is " << f(8) << endl;

return 0;

}

// pre: n >= 0

int f(int n)

{

cout << "Function entered with n = " << n << endl;

if ( (n >= 0) && (n <= 2) )

return n + 1;

else

return f(n-2) \* f(n-4);

}

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

Function entered with n = 2

Function entered with n = 0

The value of f(8) is 27

Any negative value and odd numbers would cause it to have an infinite recursion.

Infinite recursion : f(-1)

Even though the precondition states that n is nonnegative, there is no actual code to keep a negative value for n from being used as the argument in the function.

When n=3 , as 3 is not within the subrange of 0 to 2, the default will execute, thus attempting to evaluate f(1) and f(-1), which do not match with the base case.

When n is any odd number, f(n)’s arguments will be n’s next two smaller odd integers due to recursive call of f(n-2) and f(n-4), which eventually will need to evaluate f(3).