CS3021/3421 Tutorial 2

Consider the following C/C++ code segment:

```
_int64 g = 256;

_int64 p(_int64 i, _int64 j)
{
    _int64 k;
    k= i + j;
    return (k << 2) - 1;
}

_int64 q(_int64 i)
{
    return p(g, -i);
}

_int64 f(_int64 n)
{
    if (n > 0) {
        return n*f(n-1);
    } else {
        return 1;
    }
}

_int64 xp5(_int64 a, _int64 b, _int64 c, _int64 d, _int64 e)
{
    _int64 sum = a + b + c + d + e;
    printf("a = %I64d b = %I64d c = %I64d d = %I64d e = %I64d sum = %I64d\n", a, b, c, d, e, sum);
    return sum;
}
```

Q1. Translate the code segment above into x64 assembly language using the basic code generation strategy outlined in lectures.

Q2. What does the function f(n) calculate? Draw the state of the stack frames after a call to f(10) has been made during the calculation of f(13).

Q3. Using Visual Studio (or equivalent), create an x64 console application with files t2.h and t2.asm containing the x64 assembly language for p(), q(), f() and xp5(). Write C++ code to test the functions by, for example, calling f() to calculate f(1), f(2) to f(10) [see x64codegen.cpp]. Hand in listings of your code files and a screen dump of the console window showing the results of your program.

Q4. Remove the code that allocates and de-allocates the shadow space in fib64.asm. What happens when xp2 is now called from x64codegen.cpp and why?