Recursive Function Assignment:

1. WAP to calculate the maximum stack depth of a recursive call to a function. (For eg a factorial function).

```c
#include <stdio.h>
int factorial(int);
int current=0;
int max=0;
int main()
{
int n;
 printf("Enter a value:\n");
 scanf("%d",&n);
 printf("Factorial of %d is %d\n",n,factorial(n));
 printf("Maximum stack depth: %d\n",max);
}
int factorial(int n)
{
  int result;
  current++;
  if(current>max)
    max=current;
  if(n<=1)
  {
    current--;
    return 1;
  }
  else
  {
    result =n*factorial(n-1);
    current--;
    return result;
  }
}
```

```
Enter a value:
5
Factorial of 5 is 120
Maximum stack depth: 5
```

2. What is tail recursion? Why is it important? Give an example

A. Tail recursion is defined as a recursive function in which the recursive call is the last statement that is executed by the function. So basically nothing is left to execute after the recursion call. Here is an example program

```
void print(int n)
{
   if (n < 0)
      return 0 ;
   printf("%d ", n);

   // The last executed statement is recursive call
   print(n - 1);
}
```

Compilers usually execute recursive procedures by using a stack. This stack consists of all the pertinent information, including the parameter values, for each recursive call. When a procedure is called, its information is pushed onto a stack, and when the function terminates the information is popped out of the stack.

Thus ,for the non-tail-recursive functions, the stack depth (maximum amount of stack space used at any time during compilation) is more.