Recursion
M.A. El-dosuky

[1] The following Program compute 2^3

```
#include <iostream.h>
int twoRaisedTo0( ) {
        return 1;}
int twoRaisedTol( ){
        return 2 * twoRaisedTo0( );}
int twoRaisedTo2( ){
        return 2 * twoRaisedTol( );}
int twoPaisedTo3( ){
        return 2 * twoRaisedTo2( );}
int main()
{
cout « "2 to the 3rd is " «
twoRaisedTo3();
return 0 ;
}
```

* Rewrite it Using Recursion

```
int twoRaisedTo (int n ){
        if(n== 0)
          return 1;
        else
          return 2 * twoRaisedTo(n-1);
}
```

Recursive:

$$f(n)=\begin{cases}2*f(n-1) & \text{When } n>=1\\ 1 & \text{When } n=0 \text{ (BASE CASE)}\end{cases}$$

Non-recursive (closed form)

$$f(n)=2^n, \forall n \geq 0$$

Forward Substitution

$$f(5)=2*f(4)$$
$$f(4)=2*f(3)$$
$$f(3)=2*f(2)$$
$$f(2)=2*f(1)$$
$$f(1)=2*f(0)$$
$$f(0)=1$$

Backward Substitution

$$f(5)=2*f(4)=2*16=32$$
$$f(4)=2*f(3)=2*8=16$$
$$f(3)=2*f(2)=2*4=8$$
$$f(2)=2*f(1)=2*2=4$$
$$f(1)=2*f(0)=2*1=2$$
$$f(0)=1$$

[2] Explain the following:-

- Recursion :**Sometimes the best way to solve a problem is to solve a smaller version of the exact same problem first. When turn this into program, you end up with functions that call themselves.**
- Closed Form : **An equation that is not defined recursively.**
- Evaluation by substitution : **To evaluate recursive definition by hand, we keep substituting values until we get to something evaluated without recursion.**
- Recursive Search Algorithms : **the problem is to find a target in an array a of length n. To apply recursion to this problem, we need to figure out a way to solve the complete problem given a solution to a smaller version of the same problem**
- Recursive Sorting Algorithm : **the problem is to order an array a of length n. To apply recursion to this problem, we need to figure out a way to solve the complete problem given a solution to a smaller version of the same problem… Quickisort[9]**

[3] Using recursive form of factorial function (using equation-1) to substitute and evaluate f(5)

Equation 1 $$F(n)=\begin{cases} n\times F(n-1) & n\geq 1 \\ 1 & n=0 \end{cases}$$

[4] Use Equation-1 to write a program that computes the factorial function.

Answer of questions 3 and 4

Forward Substitution

$f(5)=5*f(4)$

$f(4)=4*f(3)$

$f(3)=3*f(2)$

$f(2)=2*f(1)$

$f(1)=1*f(0)$

$f(0)=1$

Backward Substitution

$f(5)=5*f(4)=5*24=120$

$f(4)=4*f(3)=4*6=24$

$f(3)=3*f(2)=3*2=6$

$f(2)=2*f(1)=2*1=2$

$f(1)=1*f(0)=1*1=1$

$f(0)=1$

```
int factorial (int n)
{
  if (n == 0)
    return 1;
  else
    return n * factorial (n - 1);
}
```

[7] Identify the errors in the following program and write the correct version;
Code for choose recursive function. The n choose k function

choose k from n

$$\binom{n}{k}=\frac{n!}{k!(n-k)!}$$

$$c(n,k)=c(n-1,k-1)+c(n-1,k)$$

$$\binom{n}{k}=\binom{n-1}{k-1}+\binom{n-1}{k}$$

$$c(n,k)=\begin{cases} n & \text{when } k=1 \\ 1 & \text{when } n=k \\ c(n-1,k-1)+c(n-1,k) & \text{when } n>k \text{ and } k>1 \end{cases}$$

```
int choose(int n, int k)
{
  if (k == 1)
    return n;
  else if (n == k)
    return 1;
  else                          // recursive case: n>k and k>1
    return choose(n - 1, k - 1) + choose(n - 1, k);
}
```