TP 2: Recursion

1. Definition :

- In programming, recursion is a method that allows a subprogram (procedure or function) to call itself.
- In the body part (instructions) we find a call to the procedure (function) itself.

Example: a C function that allows to calculate the factorial of a given integer n

Iterative solution	Recursive solution

int fact(int n)
{
int i, p=1;
 	
for(i=1;i<n;i++)
p = p * i;
 	
return p;
}
	int fact(int n)
	{
	int p;
	if (n = = 0)
	p = 1;
	else
	p = n * fact(n-1);
	
	return p;
	}

Recursive call

2. Types of recursion:

There are generally two types of recursion:

- ✓ Simple recursion
- ✓ Cross recursion

2.1. Simple recursion:

This is when a subprogram (function or procedure) calls itself. This is in fact the general case of recursion as we have already seen in the previous example with the factorial function.

Example: calculation of the sum of n first positive integers.

Sum (n) = 1 + 2 + 3 + … + (n-1) + n

```
#include <iostream>
 using namespace std;
 int Sum (int n)


{
      if (n == 1)
          return 1;
      else
          return Sum (n-1) + n;


   }

 int main()
  {
    int nbr ;
     cout <<"type an integer:"<< endl;
     cin>> nbr;
     cost << "the sum is: "<< Sum(nbr) << endl;
     getchar();
   }
```

2.2. Cross recursion:

- We call cross recursion the fact that two procedures P1 and P2 call each other, that is: when P1 executes, it calls P2, and when P2 executes, it calls P1.

Example :

- A positive integer n can be either:

  Even n = 2*k
  Odd n = 2*k+1

- If we consider two functions Even (n) and Odd (n) with logical values (boolean) then we will have:

  If Even (n) = true then Odd (n) = false
  If Odd (n) = true then Even (n) = false

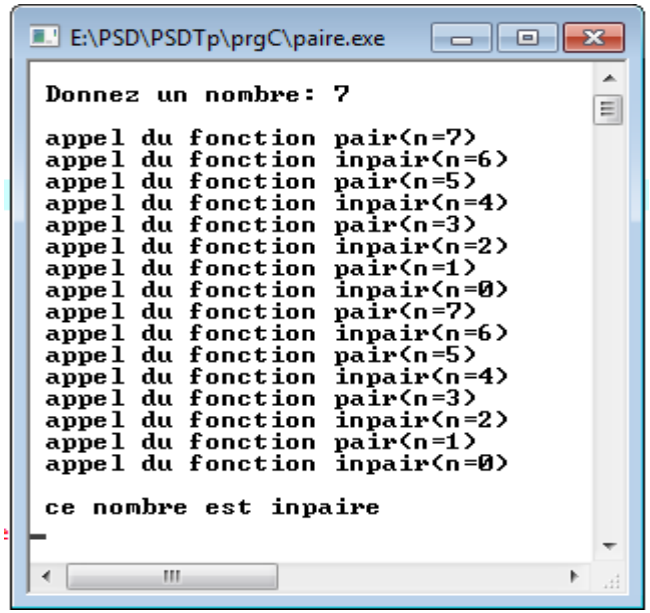
```
#include <iostream>
using namespace std;
int nbr;
// Declaration of function headers
int Even (int);
int Odd (int);
// The implementation of the functions
bool Even (int n)
 {
     cout <<" call of even function (n=" << n << ")"<<endl;
     if (n==0)
         return true;
     else
        return Odd (n-1);

 }
bool odd(int n)

 {
    cout<<" odd function call (n=" << n << ")"<<endl;
    if (n==0)
       return false;
    else
       return Eve(n-1);

  }
//main program
int main()
 {
    cout<<"Give a number: "<< endl;
    cin>>n;
   // even(nbr);
    if (even(nbr)==true)
      cout<<" this number is even "<<endl;
   else
       cout<<" this number is odd "<<endl;

  System ("pause");
  return 0;
}
```



Exercise 1:

Write a C++ program that reads an integer N and then calculates and displays the following sum: S= N! + (N-1) + . . . + 2!+1!

You must use recursive functions in your solutions.

Execice 2:

The dichotomy search for an element in an ordered array is carried out as follows:

1) We divide the table into two approximately equal parts,
2) We compare the value to be sought with the middle element,
3) If they are not equal, we only focus on the part containing the desired elements and we neglect the other part.
4) We repeat these 3 steps until we have a single element to compare.

Translate in C++ the Algorithm TP below that allows, using recursion, to read a vector of integer N and search by dichotomy for an element x in this vector.

Solution: (algorithmic)

```
Algorithme TP
Procedure ReadVe (var V:array of integers, N:integer)
Begin
If (N=1)
      Read (V[1])
Else
      Read (V[N])
      ReadVe(V, N-1);
Endif
End
```

```
// The function Search returns the rank of the element if it exists otherwise it returns -1.

Function Search (V:array of integers, Val:integer, Iinf,Isup:integer): integer
Imil: integer;
Begin

If(Iinf>Isup) then
     Return(-1); // Val does not exist in V
Else
    Imil ← (Iinf+Isup) div 2;
    if(V [Imil]= Val) then
        Return(Imil); // Val exist in V
    Else
       If(Val< V [Imil]) then
             Return (Search (V, Val, Iinf, Imil-1));// rec
       Else
         Return (Search (V, Val, Imil+1, Isup));//rec
       Endif
    Endif
Endif

END

T[10]: Array of integers
N, Val, rank: integer;
Begin
 Read (N);
 ReadVe (T, N);
 Rank ← search (T, Val, 1, N); // in C++ search (T, Val, 0, N-1);
 Write (Rank);
End
```