Recursive Functions & Pointers

Outline

- **Recursive Functions**
- **Pointers**

Recursive Functions

As it was said before modules in C are called functions. One of the types of functions is a recursive function. A recursive function is a function that calls itself for further simplified processing. As the function calls itself the task is divided into more manageable subtasks. The recursive function calls continuously till a known base-case is found. A typical recursive function has a recursive formula and a base case.

Example: Consider the factorial problem:

n!=n*(n-1)! **(Recursive formula)**
1!=1*(0!) **(Base-case)**

The recursive function that can be written for calculating a recursive function is as follow:

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

Iteration and recursive functions, both are doing the same. In a recursive problem there are two important points which needs to be considered:

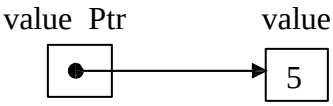
1. **Recursive formula**
2. **Base-Case**

1. Write a C program to calculate the result of find power of a number *x* to the base *a*

(***HINT:*** *The recursive formula is:*
*power(x,a)=x*power(x,a-1)*
and the base case is:
power(x,0) = 1).

Pointers

Pointers are variables that contain memory addresses as their values. Normally, a variable directly contains a specific value, but pointer contains an address of a variable that contains a specific value.



Pointers must be declared before they can be used. The declaration: **int** *value_Ptr, value;
declares the variable **value_Ptr** to be of type **int *** (pointer to an integer value) and variable **value** to simply be an integer.
When ***** is used in this manner in a declaration it indicates that the variable being declared is a pointer. Pointers can be declared to point to objects of any type.

Pointer Operators

The **&**, or *address operator* is a unary operator that returns the address of its operand. Example:
Assume two declarations below:

int *x=4, *xPtr;* and statement : *xPtr = &x;*

assigns the address of the variable **x** to pointer **xPtr**. Variable **xPtr** then points to **x**.
The ***** operator, called *indirection operator* or *dereferencing operator* returns the value of the object to which its operand (i.e. pointer) points. Following previous example, statement :

printf*("%d", *xPtr);*

prints the value of variable **x** (which is 4).
The **&** and ***** are complements of one another.
Following example shows usage of the pointer operators.

```
#include <stdio.h>
int main()
{
   int a;
   int *aPtr;
   a = 7;
   aPtr = &a;
   printf( "\n The address of a is %p\n and the value of *aPtr is %p", &a, aPtr);
   printf( " \n The value of a is %d\n and the value of *aPtr is %d", a, *aPtr);
   printf( " \nShowing that * and & are inverses of each other. \n  &*aPtr = %p and *&aPtr = %p
\n",
           &*aPtr, *&aPtr);
}
```

Arrays and Pointers

In C, pointers and arrays are related. An array is actually a pointer to the 0th element of the array. Dereferencing the array name will give the 0th element. This gives us a range of equivalent notations for array access. In the following examples, arr is an array.

Array Access	Pointer Equivalent
arr[0]	*arr
arr[2]	*(arr + 2)
arr[n]	*(arr + n)

To understand the relationship between pointers and arrays, investigate the following function.

```
float  avg (float *a, int size)
{
int i;
float sum; sum=0;
for(i=0;i<size;i++)
        sum+=*(a+i);
return (sum/size);
}
Call this function inside main.
#include<stdio.h>
#define SIZE 10
float  avg(float *a, int size);
main(){
        float a[SIZE] ,average;
        for (int i=0;i<SIZE;i++)
        scanf("%f",&a[i] );
        average= avg(a,SIZE );
        printf("The average of  the %d numbers is %f",SIZE,average);
       }
```

2.a (Due one week)

An integer is said to be prime if it is divisible only by two distinct factors 1 and itself. For example, 2, 3, 5, and 7 are prime, but 4, 6, 8, and 9 are not. [*Note*: The number 1 is not a prime number]. Write a function that determines if a number is prime. Use this function in a program that determines and prints prime numbers between 1 and 1000. Use pointer approach to implement the program.

2.b (Due one week)

Write a program that takes an integer number with four digits and returns it with its digits reversed. For example, given the number **7631**, the program should return **1367**. Use at least function one function and use pointers in the program.

NOTES:

1) Please Send your Homework in the following Emails, but remember who is your **lab instructor**

emu.clab2@gmail.com for Pouya's Student
eenglab212@gmail.com for Mohamad's Student

2) Subject of email Should include student Number + Homework Number
For Example: "St. 15000012 Homework #1"

3) Your homework should be saved with your student number and attached as notepad.
For Example 15000012.txt