Week 11 Lab

1. The Ackermann function is defined as follows:week1
   - If *m* is 0, A(m,n)=n+1.
   - If m>0 and n=0, then A(m,n)=A(m−1,1).
   - If m>0 and n>0, then A(m,n)=A(m−1,A(m,n−1)).

   Write a recursive C function **ackermann** that returns int and accepts two integers to compute the Ackermann function A(m,n) given two non-negative integers m and n.
2. Below is a C function that returns 1 if the given string containing only parentheses is balanced. Study it to write its recursive version.

```c
int is_balanced(char str[]) {
    int count = 0;

    for (int i = 0; str[i] != '\0'; i++)
    {
        if (str[i] == '(')
        {
            count++;
        }
        else if (str[i] == ')')
        {
            count--;
            if (count < 0)
            {
                return 0;
            }
        }
    }

    return count == 0;
}
```

Notice that the variable "i" is responsible for tracking the current character in the string, and at each iteration, the value of str[i] is examined. Consequently, your function should incorporate "i" as one of its parameters.

Similarly, the variable "count" is incremented upon encountering an opening parenthesis '(' and decremented when a closing parenthesis ')' is found. The code effectively monitors the balance of parentheses through the "count" variable. It is imperative to adhere to this mechanism. Therefore, in your recursive function, "count" should also be included as a parameter.

The general approach is as follows: increment "count" when an opening parenthesis '(' is encountered, signifying the count of '(' characters seen. Conversely, when a closing parenthesis ')' is found, decrement "count" to cancel a ')' that has been previously encountered. It is crucial to check that "count" does not go below 0 during the decrement operation, as this situation indicates an unpaired ')' without a matching '('—a condition where "count" becomes negative.

The final line ensures that "count" is 0. At this point, it could be either 0 or a positive number (considering that we return 0 if "count" becomes negative).