**Lab 3 – Observing Algorithms**

1. **Tower of Hanoi**

Code the ToH.

1. Print the moves for a user input number of disks.

2. Each recursive call to the ToH places the ToH function on the stack. Print the stack level of each call (you can use a global variable for this, or pass it as an argument to the function)

1. **3-sum**

3-Sum. Given *N* distinct integers, how many triples sum to exactly zero?

e.g. Given an array of { 30, -40, -20, -10, 40, 0, 10, 5 }

How many triples of these numbers sum to 0?

|  |  |  |  |
| --- | --- | --- | --- |
| **a[i]** | **a[j]** | **a[k]** | **sum** |
| **30** | **-40** | **10** | **0** |
| **30** | **-20** | **-10** | **0** |
| **-40** | **40** | **0** | **0** |
| **-10** | **0** | **10** | **0** |

* Write a program that takes the above input and then prints out how many triples summing to 0 there are.
* In the example above, the program should print out 4

1. **Timing**

Time how long your algorithm takes to run. Use the C timing library to record the start and end of your program’s run time – the actual run time is the difference between the end and the start.

1. Include the <time.h> library,
2. Declare two variables – for the start and end time values
3. Start the clock – assign the value of the clock() function to the start
4. End the clock – capture the clock() return
5. Calculate the difference between them for the elapsed time.
6. Display the result. Note the result is cast from time\_t to double.
7. The result is in clock ticks – to display seconds, divide by the architecture’s value for clock ticks per second – CLOCKS\_PER\_SEC. (this is a #define in the time.h file)

The lab section has a file which contains 1999 numbers!! You can use the text in your file to initialise your array if you’d like to test it for bigger numbers. Just put int A[]=<paste contents of file>; If you like, test and time your code for different values of A e.g. 100, 200, 400, 800 etc – edit the loop ending variable to suit.

int main()

{

time\_t pStart,pEnd; //time\_t is a variable type

pStart = clock(); //Record start time

/\* do your code \*/

pEnd = clock(); //record end time

printf (

"Elapsed time is %1.7lf seconds\n",

(double)(pEnd-pStart)/CLOCKS\_PER\_SEC);

}

1. **fibonacci series**

Write the recursive code for fibonacci. Time it for values 10, 20, 30, 40 & if you have the time, 50.