Question 1:

(a) Average cycles-per-second performance for the fibonacci.c program is ***25455.2272727***
(b) Average cycles-per-second performance for the gregoryleibniz.c program is ***32564.3939394***
(c) Average Cycles-per-Instruction (CPI) for the fibonacci.c program is ***2.61333443633***
(d) Average Cycles-per-Instruction (CPI) for the gregoryleibniz.c program is ***4.26734835699***
(e) Average number of memory transactions per cycle for the fibonacci.c program is ***0.666292217822***
(f) Average number of memory transactions per cycle for the gregoryleibniz.c program is ***0.338466907061***
(g) Discrepancies between the Fibonacci & Gregory-Leibniz in (a)-(f):

Average CPI: (c,d)
The average CPI of is dependent on the compute intensity, the Gregory Leibniz algorithm has a higher average CPI comparatively owing to its high compute intensity due to floating point operations involved.
Average number of memory transaction per cycle: (e,f)
The average number of memory transactions in the fibonacci algorithm is higher comparatively due to use of recursive function which leads to memory accesses while the gregory-leibniz algorithm doesn't involved very frequent memory accesses comparatively.
Average cycles-per-second performance:
The average cycles-per-second performance of the Gregoryleibniz.c program seems to be better than the Fibonacci.c program. This performance may be owed to frequent memory accesses in the fibonacci algorithm, memory access latency is usually very high.

Question 2:

Determining the 32 bit address for the transaction: (AHB Lite)

S.No	Chip Select (1 bit)	Row Select (13 bits)	Bank Select (2 bits)	Column Select (12 bits)	Byte Select (4 bits)	**Address(Hex) (8 hex)**
1	0	0000000000000	00	000000000001	0000	0x00000010
2	0	0000000000000	00	000000010000	0000	0x00000100
3	0	0000001000000	00	000000010000	0000	0x01000100
4	0	0000001000000	01	000000010000	0000	0x01010100

S.no 1., S.no 2: The first and second read to the same row and bank
S.no 3: Read to a different row (same bank)
S.no 4: Read to a different bank.

16 bytes of data to be transferred for each read. Each byte needs a select line, 2^4 = 16. Hence the last 4 bits must be used as 0000 to access the whole 16 bytes.

Latency Expected for this transaction:

Cas latency can be excluded if the pipeline is ideal but the following calculation is including Cas latency.

AHB Data width = 4 bytes. So for 16 bytes, 4 burst transaction is required.

Start of Transaction 1 to Start of transaction 2
Transaction 1 → 2 expected latency:
Expected latency is 0 as the same row and same row.
Transaction 2 → 3 expected latency:
t_rp + t_rcd + t_cas = 3 + 2 + 2 = 7
Change in row : precharge delay should be added, cas and rcd must also be added
Transaction 3 → 4 expected latency:
t_rrd + t_rcd + t_cas = 0 + 2 + 2 = 4

Latency Simulated for this transaction:

Considering the latency as the difference of time from end of one read instruction to the start of the next.

Latency for read instruction 1 - 2: (Same Row and Same Bank)
= 60ns = 6 cycles

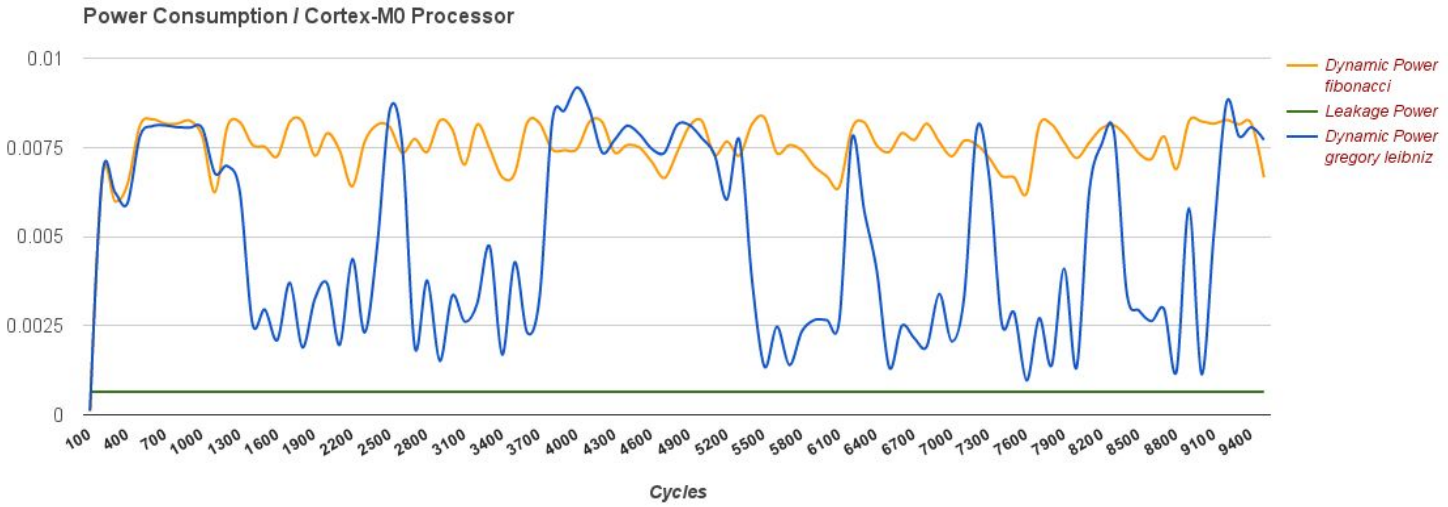
Latency for read instruction 2 - 3: (Different Row and Same Bank)
= 110 ns = 11 cycles

Latency for read instruction 3 - 4: (Same Row and Different Bank)
= 110 ns = 11 cycles

End of transaction I to Start of transaction II I → II	Expected Latency	Observed Latency	Discrepancy
1 → 2	0	6	6
2 → 3	7	11	4
3 → 4	4	11	7

A discrepancy is observed as additional internal delays may be added in the pipeline.

Question 3:



The Static Leakage Power observed during simulation of both the algorithms is the same owing to using the same RTL simulations of the Cortex M0 processor.
Whereas the Dynamic Power is dependent on the algorithm, based on factors such as the number of times memory is accessed, compute intensity.

Gregory-Leibniz algorithm:

The Gregory-Leibniz algorithm for calculating the value of pi includes the excessive use of floating point operations whereas the memory accesses are to the heap memory and not the Memory.
Sharp changes observed in the graph may be due to change in computation in different parts of the program.

Fibonacci Algorithm:

Fibonacci algorithm given uses recursive functions. Fib(i) is a recursive function. This algorithm involves frequent memory accesses, hence the power consumed is constantly high.

Comparing the two algorithms, the Gregory-Leibniz algorithm seems to consume lesser power on an average comparatively as the fibonacci algorithm consumes constantly high power due to persistent memory accesses. However the two algorithms cannot be compared in terms of performance as the two algorithms deal with different issues.