Solution to Question 1
==========

Main memory size = 64K bytes
Cache size = 1K bytes
Block size = 128 bytes, 128bytes are fetched together into the cache while replacement.

There are 1K/128 = 8 sets in the cache, which is direct-mapped.

Memory is byte addressable, while each word is 4bytes, each instruction would be of 1word length, so accesses would be in fashion,
0-3, 4-7, and so on. Whereas 128 bytes of consecutive data would be fetched together into direct mapped cache.

Starting the loop:

Request, Event,	Time(in multiples of Tau)
Access to 8 - Cache miss	81
All subsequent accesses upto 127 is brought in set 0 of cache.	
Access 12-55, 11 accesses to same cache set	11
	92
Loop 1: first part	
56-127, 18 instruction accesses, cache hit	18
Access 128 - cache miss	81
All subsequent access upto 256 is brought in set 1 of cache	
Access 132 - 136, cache hit	2
	101
Loop 2:	
Access 140-240, cache hits	26
Loop2 executes 20 times, Total time	500
Loop 1: Last part	
Access 244 - 252, cache hit	3
Access 256, cache miss, bringing 256 - 383 in set 2	81
Access 260 -380, cache hit	31
Access 384, cache miss, bringing 384-511 in set 3	81
Access 388 - 508, cache hit	31
Access 512, miss, bringing 512 - 636 in set 4	81
Access 516-636, cache hit	31

Access 640, miss, bringing 640-767 in set 5	81
Access 644-764, cache hit	31
Access 768, miss, bringing 768-895 in set 6	81
Access 772-895, cache hit	31
Access 896, miss, bringing 896-1023 in set 7	81
Access 900-1020, cache hit	31
Access 1024, miss, bringing 1024-1151 in set 0	81
Access 1028-1148, hit	31
Access 1152, miss, bringing 1152 - 1279 to set 1	81
Access 1156-1200, hit	2

—----------

9*81+ 7*31 + 3+ 2 = 951

First Run of Outer Loop takes: 101 + 500 + 951 = 1552
From second Run, there are four more cache misses due to replacements of set 0 and set 1,
Rest all are cache hits, as all elements have been loaded into the cache already
4*80 + (21+ 25*20+ 240)(cache hits) = 1081

Now this particular run will go for 9times = 1081*9 = 9729
—----------

End part:

Access 1200 - 1276, cache hit,	20
Access 1280, miss,	81
Access 1284- 1407, cache hit	31
Access 1408, miss	81
Access 1412-1504	23
	—----------
Total:	236

Total Execution Time =
92 + 1552 + 9729 + 236 = 11609
==========

Solution to Question 2a:

Multiply +25 and -19 using Booth's algorithm

25: 011001 → Q
-19: 101101 —> M
19: 010011 —-> -M

	A	Q	Q-1
	000000	011001	0
-M	010011		
A-M	010011	011001	0
Shift	001001	101100	1
——	----------	----------	-
M	101101		
A+M	110110	101100	1
Shift	111011	010110	0
——	----------	----------	-
Shift	111101	101011	0
——	----------	----------	-
-M	010011		
A-M	010000	101011	0
Shift	001000	010101	1
——	----------	----------	-
Shift	000100	001010	1
——	----------	----------	-
M	101101		
A+M	110001	001010	1
Shift	111000	100101	0
——	----------	----------	-

Final Result: 111000 100101

==========

Solution to Question 2a:

Delay of 2-input NAND = 5ns
Delay of 2input NOR = 5ns
Delay of NOT = 3ns

Delay of a 6 bit ripple carry adder
——----------
Delay of a Full Adder:
Carry: A.B+A.C+B.C
Sum: A xor B xor C

Equation of the sum output for the full adder circuit with NAND gates isobtained as follows –

$$S = (A \oplus B) \cdot \overline{(A \oplus B) C_{in}} \cdot C_{in} \cdot \overline{(A \oplus B) C_{in}} = A \oplus B \oplus C_{in}$$

Where,

$$A \oplus B = \overline{A \cdot \overline{AB} \cdot B \cdot \overline{AB}}$$

And equation of the carry output of the full adder circuit withNAND gate is given by,

$$C_{out} = \overline{C_{in} (A \oplus B) \cdot \overline{AB}} = AB + (A \oplus B) C_{in}$$

General Solution to the problem:
—----------

Say, one Full Adder has been implemented using NAND, NOR and NOT gates,
Where Sum is implemented using x1 NAND, y1 NOR and z1 NOT gate
And carry using x2, y2, and z2 gates respectively.

Delay of FA sum: 5(x1+y1) + 3z1
Delay of FA Carry: 5(x2+y2) + 3z2

Now the ripple through 6 bit ripple carry would be,

Final Carry_out generated after: 6*(5x2+5y2+3z2)
All 6 bits of Sum available after: 5*(5x2+5y2+3z2) + (5x1+5y1+3z1)

I donot expect the students to write this expression properly,
If they can do the basic propagation steps of ripple carry adder, with certain assumptions
Of the sum and carry, then please go ahead with awarding 80% of the marks.