Q: Are all of these enough to get full marks in the exam?

A: NO. This is a practice sheet. Meaning, you can practice all you want using the questions from this sheet. However, doing well in exams depends upon your ability to understand a question, formulate an answer, and express it correctly. You see, these are humane skills which cannot be guaranteed by completing a practice sheet only. But yeah, Best of luck anyway.

Chapter 2 (Instructions: Language of the Computer)

Question - 1:

Construct the equivalent RISC-V code of the following C code. Once you have the RISC-V code, identify type of each instruction and encode them accordingly.

```
A[7] = A[2] + A[B[8]] + 10;
B[i] = A[3] - 8;
```

Base addresses of array A and B are in register X_{20} and X_{21} and i is in register X_{22}

Question - 2:

Construct the equivalent RISC-V code of the following C code.

```
for (i = 8; i > 0 ; i--) {
  if ( A[i] == i){
     A[2] = A [B[3]] ;
  }
}
```

Base addresses of array A and B are in register X_{20} and X_{21} . Also consider i is in register X_{22}.

Question - 3:

Construct the equivalent RISC-V code of the following C code.

```
if ( A[i] < i){
     A[2] = A [B[3]] ;
}
```

Base addresses of array A and B are in register X_{20} and X_{21} . Also consider i is in register X_{22}.

Question - 4:

Construct the equivalent RISC-V code of the following C code.

```
if ( A[3] != A[6]){
     if (A[3] == 0) {
          A[3] = A[3] + 2;
     }else{
          A[6] = A[6] / 16;
     }
}else{
     A[6] = A[6] * 8
}
```

Base addresses of array A and B are in register X_{20} and X_{21} .

Question - 5: (Skip)

Construct the equivalent RISC-V code of the following C code.

```
Main () {
    int x = 0;
    int y = 9;
    int z = addition(x, y);
}

int addition (int a, int b) {
```

```
    int c = a + b;
    return c;
}
```

Variables x, y, z are stored in X_{20}, X_{21} and X_{22} registers. Argument x, y are passed using register X_{13}, X_{14}
Variable c from the addition function also uses register X_{21}

Question - 6:

Write RISC-V assembly code that checks if the number stored in register $\mathbf{X_{25}}$ is **even** or not. If **even** then store **1** in register $\mathbf{X_{26}}$ otherwise store **0**.

Question - 7:

ADD X_{25} , X_{25} , X_0 . Can you make this instruction faster? If yes, Write the updated instruction?

Question - 8:

Memory Location	Code	Line Number	Machine Code
	ADDI $\mathbf{X_5}$, X_0 , 5	1	
	ADDI $\mathbf{X_6}$, X_0 , 1	2	
	ADDI $\mathbf{X_{25}}$, X_0 , 0	3	
	Loop: BLT $\mathbf{X_5}$, $\mathbf{X_6}$, loopBreak	4	_____XXX_____XXXXXXX
	ADDI X_{25} , X_{25} , 1	5	
#7080	ADDI X_5 , X_5 , -1	6	
	BEQ $\mathbf{X_0}$, $\mathbf{X_0}$, Loop	7	_____XXX_____XXXXXXX
	loopBreak:	8	

a) What is the value of **PC** while executing line2? Answer: _____
b) Fill up the machine codes corresponding to line4 and line7 in the table above.

Question - 9:

Memory Location	Code	Line Number
	Loop:	
	SLLI $\mathbf{X_{10}}$, X_{22} , 3	1
	ADD $\mathbf{X_{10}}$, X_{10} , X_{25}	2
	LD $\mathbf{X_9}$, 0(X_{10})	3
	BNE $\mathbf{X_9}$, $\mathbf{X_{24}}$, Exit	4
#80016	ADDI X_{22} , X_{22} , 1	5
	BEQ $\mathbf{X_0}$, $\mathbf{X_0}$, Loop	6
	Exit:	

a. Fill up the memory locations.
b. Find the SB-type instructions from the above code and encode them accordingly. Given,
   I. opcode = $(103)_{10}$, funct3 = $(000)_2$ opcode for BEQ
   II. opcode = $(103)_{10}$, funct3 = $(001)_2$ opcode for BNE

Question - 10:

Write necessary RISC-V instructions to store the value (1111 1111 0000 1111 11)2 in X20 register.

Question - 11:

Show how the value 0xabcdef12 would be arranged in memory in RISC-V machine.

Question - 12:

For the RISC-V assembly instructions below, what is the corresponding C/high level statement?

slli x30, x5, 3
add x30, x10, x30
slli x31, x6, 3
add x31, x11, x31
	Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7,

ld x5, 0(x30)
addi x12, x30, 8
ld x30, 0(x12)
add x30, x30, x5
sd x30, 0(x31)
	x28, and x29, respectively. Assume that the base address of the
	
	Arrays A and B are in registers x10 and x11, respectively.