Your Name (please print clearly) _____

This exam will be conducted according to the Georgia Tech Honor Code. I pledge to neither give nor receive unauthorized assistance on this exam and to abide by all provisions of the Honor Code.

Signed _____

1	2	3	4	total
30	20	25	25	100

Instructions: This is a closed book, closed note exam. Calculators are not permitted.

Read each question over before you start to work. If you need to make any assumptions, state them. If you have a question, raise your hand; do not leave your seat. The meaning of each question should be clear, but if something does not make any sense to you, please ask for clarification.

Please work the exam in pencil and do not separate the pages of the exam. If you run out of room, please continue on the back of the previous page. For maximum credit, show your work.

Good Luck!

Problem 1 (2 parts, 30 points) **Loops**

Part A (12 points) Given an array `int A[100]` of **non-unique** integers **sorted** in increasing order, complete the following code fragment which computes the mode of the data in A. Note that each element in the array is a member of a run whose length is greater than or equal to one.

```
int   A[100]   =   {-42,22,22,75,75,75...121};   //   given
int i;
int ThisNum,Mode;
int ThisNumCnt,ModeCnt;

Mode=ThisNum=A[0];
ModeCnt=ThisNumCnt=0;

for (i=0;i<100;i++){
     if (ThisNum == A[i])
          ThisNumCnt++; // increment run length of ThisNum
     else {

          _____________________________________________; //A.1

          _____________________________________________;
     }
     if (ThisNumCnt >= ModeCnt) {

          _____________________________________________; //A.2

          _____________________________________________;

     }


}
printf("The mode is %d with %d occurrences\n",Mode,ModeCnt);
```

A.1 Update ThisNum and ThisNumCnt for the next distinct value in the array A.
A.2 Update Mode and ModeCnt.

Part B (18 points) Write MIPS code for the for loop fragment in Part A. Assume ThisNum, Mode, ThisNumCnt and ModeCnt are in registers $4, $5, $6, and $7 respectively. Also assume that the address of array A is in $10. Your loop control variable should be in $1. You may assume all variables are initialized as given. *For maximum credit use a minimum number of instructions.*

Label	**Instruction**	**Comment**
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#

Problem 2 (2 parts, 20 points) **Conditionals: Nested if-then-else**

Consider the following MIPS code:

```
.data
ALoc: .word 10
BLoc: .word 20
CLoc: .word 30
DLoc: .word 30
Result: .alloc 1
.text
       lw   $1, ALoc($0)
       lw   $2, BLoc($0)
       lw   $3, CLoc($0)
       lw   $4, DLoc($0)
       ori  $5, $0, 1
       slt  $6, $3, $4
       bne  $6, $0, Write
       beq  $1, $2, Write
       addi $5, $0, 0
Write: sw   $5, Result($0)
       jr   $31
```

Part A (8 points) What is written to the memory location labeled Result by this program?

Result: _____

Part B (12 points) Suppose the labels ALoc, BLoc, CLoc, DLoc, and Result are the addresses of the C variables A, B, C, D, and R. What C expression is being computed by this code? Write your C-code in terms of A, B, C, and D and use logical predicates.

R = _____;

Problem 3 (4 parts, 25 points) **Assembly Programming**

Part A (6 points) Suppose L1 = 0x00234500 in the original code below. Write the **sequence** of MIPS instructions that is necessary to achieve the intent of the original instruction.

Original:	Equivalent MIPS instructions:
`addi $3, $0, L1`	

Part B (6 points) Write a **single** MIPS instruction that is equivalent to the original fragment. Assume big *endian* byte ordering.

Original:	Equivalent MIPS statement:
`lui $4, 0xFF00`	
`lw $3, 1000($0)`	
`and $3, $3, $4`	
`sra $3, $3, 24`	

Part C (7 points) Write a MIPS fragment with 1 **instruction** that is equivalent to the original fragment under a certain condition.

Original:	Equivalent MIPS in **one** instructions only:
`bne $1, $2, next`	
`j Target`	
`next:`	

Under what condition are the fragments equivalent? _____

Part D (6 points) What hexadecimal value will be in register $2 when this MIPS fragment executes? Assume big *endian* byte ordering.

```
.data
In: .word 0xABCD1234
.text
    addi $3, $0, In
    lbu  $2, 2($3)   # note this is lbu
```

Problem 4 (2 parts, 25 points) **Assembly Programming**

Part A (12 points) What does the following code fragment print?

```
int i, x;
int A[10] = {99, 33, 44, 22, 56, 78, 1, 5, 9, 88};

for(i=0; i<10; i=i+2){
  if(i & 2)              // bitwise AND
     continue;           // and don't print x
  x = A[i];
  printf("x = %d\n", x);
}
```

Fill in the blanks to rewrite the code above to produce the equivalent behavior without using `continue`.

```
int i;
int A[] = {99, 33, 44, 22, 56, 78, 1, 5, 9, 88};

for(_____; _____; _______){
  x = A[i];
  printf("x = %d\n", x);
}
```

Part B (13 points) Answer the three questions below about the following C fragment.

```
int i, j, k, count;
j = k = count = 0;

for (i=0; i < 9; i++){                    // outer loop
   if (i % 3) continue;
   j = 0;
   while (j < 9){                         // middle loop
     k = 0;
     while (k < 9){                       // inner loop
       if (k % 3) break;
       k++;
     }
     count++;
     j++;
   }
}
printf("%d\n", count);
```

How many times is `break` executed?
How many times is `continue` executed?
What is printed?

MIPS Instruction Set (core)

instruction	*example*	*meaning*
arithmetic		
add	add $1,$2,$3	$1 = $2 + $3
subtract	sub $1,$2,$3	$1 = $2 - $3
add immediate	addi $1,$2,100	$1 = $2 + 100
add unsigned	addu $1,$2,$3	$1 = $2 + $3
subtract unsigned	subu $1,$2,$3	$1 = $2 - $3
add immediate unsigned	addiu $1,$2,100	$1 = $2 + 100
set if less than	slt $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if less than immediate	slti $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
set if less than unsigned	sltu $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if < immediate unsigned	sltui $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
multiply	mult $2,$3	Hi, Lo = $2 * $3, 64-bit signed product
multiply unsigned	multu $2,$3	Hi, Lo = $2 * $3, 64-bit unsigned product
divide	div $2,$3	Lo = $2 / $3, Hi = $2 mod $3
divide unsigned	divu $2,$3	Lo = $2 / $3, Hi = $2 mod $3, unsigned
transfer		
move from Hi	mfhi $1	$1 = Hi
move from Lo	mflo $1	$1 = Lo
load upper immediate	lui $1,100	$1 = 100 x 2^{16}
logic		
and	and $1,$2,$3	$1 = $2 & $3
or	or $1,$2,$3	$1 = $2 | $3
and immediate	andi $1,$2,100	$1 = $2 & 100
or immediate	ori $1,$2,100	$1 = $2 | 100
nor	nor $1,$2,$3	$1 = not($2 | $3)
xor	xor $1, $2, $3	$1 = $2 ⊕ $3
xor immediate	xori $1, $2, 255	$1 = $2 ⊕ 255
shift		
shift left logical	sll $1,$2,5	$1 = $2 << 5 (logical)
shift left logical variable	sllv $1,$2,$3	$1 = $2 << $3 (logical), variable shift amt
shift right logical	srl $1,$2,5	$1 = $2 >> 5 (logical)
shift right logical variable	srlv $1,$2,$3	$1 = $2 >> $3 (logical), variable shift amt
shift right arithmetic	sra $1,$2,5	$1 = $2 >> 5 (arithmetic)
shift right arithmetic variable	srav $1,$2,$3	$1 = $2 >> $3 (arithmetic), variable shift amt
memory		
load word	lw $1, 1000($2)	$1 = memory [$2+1000]
store word	sw $1, 1000($2)	memory [$2+1000] = $1
load byte	lb $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
load byte unsigned	lbu $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
store byte	sb $1, 1002($2)	memory[$2+1002] = $1 (byte modified only)
branch		
branch if equal	beq $1,$2,100	if ($1 = $2), PC = PC + 4 + (100*4)
branch if not equal	bne $1,$2,100	if ($1 ≠ $2), PC = PC + 4 + (100*4)
jump		
jump	j 10000	PC = 10000*4
jump register	jr $31	PC = $31
jump and link	jal 10000	$31 = PC + 4; PC = 10000*4