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	24	21	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!

http://cliparts.co/bicycle-clip-art-free

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

Part A (12 points) Given an array `int A[100]` of **unique** unsorted integers and an integer `j` where `0 <= j < 100`, write a C fragment to calculate the `index (0 <= index < 100)` that element `A[j]` would have in the array if the array were sorted from smallest to largest. ***For maximum credit, declare and initialize any necessary variables.***

```
int j = … ;  // given

int A[100] = {22,-41,10001,...42}; // given
int index;
```

Part B (18 points) Write MIPS code for the fragment in Part A. **Assume `j` is given in register $1. Store the `index` computed in register $2**. *For maximum credit use a minimum number of instructions.*

Label	Instruction	Comment
	.data	#
A:	.word 22, -41, 10001, …, 42	# given set A
	.text	#
	addi $1, $0, …	# given j
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#
		#

Problem 2 (2 parts, 24 points) **Conditionals: Compound Predicates**

For the following MIPS code, assume that $1, $2, $3, and $4 are assigned to integers x, y, z and w respectively.

```
start:      bne   $1, $0, L1
            slti  $5, $2, 10
            bne   $5, $0, L2
            slt   $5, $3, $4
            beq   $5, $0, L2
L1:         add   $1, $3, $4
      j     end
L2:         add   $1, $2, $3
end:
```

Part A (12 points) Draw the control flow graph for the MIPS code shown.

Part B (12 points) Write the C code that corresponds to the above MIPS code with only one `if` statement (not nested) and only one compound predicate.

Problem 3 (4 parts, 21 points) **MIPS Equivalences**

Part A (3 points) Write a **single** MIPS instruction that is equivalent to the original fragment.

Original:	Equivalent MIPS statement:
`ori $3, $0, SetA`	
`add $4, $3, $1`	

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

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

Part C (6 points) Write a MIPS fragment with **at most 2 instructions** that is equivalent to the original fragment.

Original:	Equivalent MIPS in **two** instructions only:
`slt $3, $2, $1`	
`bne $3, $0, Target`	
`beq $1, $2, Target`	

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

```
lui $1, 0xABCD
ori $1, $1, 0x1234
sw  $1, 1000($0)
lb  $2, 1002($0)    # note this is lb, not lbu
```

Register $2:

Problem 4 (2 parts, 25 points) **Nonlocal Control Flow**

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++){
  if(i & 1)              \\ bitwise
     continue;
  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 A[4] = {1, 10, 100, 1000};
int B[4] = {2, 4, 8, 16};
int i, j, k;

for (i = 0; i<4; i++)                          \\ outer loop
   {
    for (j = 0; j<4; j++)                      \\ middle loop
       {
          if (j == 2)
           break;

          for (k = 0; k<4; k++)                \\ inner loop
           {
              if (k == 1)
                 continue;
              printf("%d\n", A[i]*B[k]);
           }
       }
   }
```

How many times is `break` executed?
How many times is `continue` executed?
How many `printf` statements are executed?

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