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	25	20	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 (10 points) The array `Temps` holds a set of 200 temperatures, represented as integers. Write a **for loop** in C that computes the average of the boiling temperatures in `Temps` (those greater than or equal to 100) and stores the average in the variable `Avg` (ignore any remainder). Assume there is at least one boiling temperature in `Temps`. *For maximum credit, declare and initialize variables as needed.*

```
int Temps[200] = {4, -1, 103, …, -36, 125};
int Avg;
```

Part B (20 points) Write a MIPS code fragment that is equivalent to the code you wrote in Part A. **Store the average in $1**. *For maximum credit, include comments and use a minimal number of instructions.*

Label	Instruction	Comment
	.data	
Temps:	**.word 4, -1, 103, …, -36, 125**	**# alloc & init Temps[200]**
	.text	
AvgBoiling:		

Problem 2 (3 parts, 25 points) **Conditionals & Compound Predicates**

For this problem, assume these registers hold the values of these variables:

$2: Answer	**$4: Hs**	**$6: He**	**$7: Ss**	**$9: Se**	**$10: temp**	**$11: Max**

Part A (9 points) Write a MIPS code fragment that computes the maximum of `Ss` and `Hs` and stores it in `$11`. Use `$10` as a predicate register. Use a minimal number of additional registers as needed. *For maximum credit, include comments.*

Label	Instruction	Comment

Part B (10 points) Consider the following MIPS code fragment.

Label	Instruction
	`slt $10, $9, $4`
	`bne $10, $0, Below`
	`slt $10, $6, $7`
	`bne $10, $0, Below`
	`add $2, $0, $11`
	`j End`
`Below:`	`addi $2, $0, 0`
`End:`	`...`

What is the equivalent C code fragment in terms of `Hs, He, Ss, Se, Max,` and `Answer`? For maximum credit, use a compound logical predicate wherever possible. Assume the variables are all of type `int`.

Part C (6 points)

If `Ss` and `Se` are integers representing years on a timeline, where `Ss < Se,` and `Hs` and `He` are years on another timeline where `Hs < He`, draw an example of a case where `Answer = 0 ($2=0)` computed in the code in Part B (select values for `Hs, He, Ss,` and `Se` to illustrate this case).

Problem 3 (2 parts, 20 points)	**MIPS Controller and Instructions**

Part A (8 points) Suppose the instruction "`jal Foo`" is executed which changes the values of the following registers to:

Register	Value
$31	2032
PC	2056

What is the address of the first instruction of the subroutine `Foo` and what is the address of the `jal Foo` instruction?

Subroutine `Foo` starts at address: ______________________

Address of `jal Foo` instruction: ______________________

Part B (12 points) For each of the following, write a single MIPS instruction to implement the C fragment? Assume variables `A, B, C,` and `D` are of type `int` and are stored in registers `$1, $2, $3,` and `$4`.

`A = 0xAB020000;`	
`B = C & 3;`	
`C = D / 512;`	

Problem 4 (2 parts, 25 points) **More Loops and Conditionals**

Part A (15 points) Suppose `A` is an array of 100 integers that might contain duplicate elements and `x` and `position` are variables of type integer. Write a **do while** loop that determines whether `x` is an element of `A` and if so, sets `position` to the smallest index of `A` at which `x` appears. Otherwise, if `x` is not in `A`, it sets `position` to `-1`. Declare and initialize any additional variables you need. For full credit, **do not** use the **`break`** statement.

Part B (10 points) What does the following code fragment print?

```
int A[10] = {1990, 3, 1992, 5, 1999, 1, 2001, 3, 2005, 9};
int B[10] = {1991, 2, 1993, 3, 1998, 7, 2002, 3, 2007, 6};
int i,j;
for(i=1; i<10; i=i+2)
  {
    for (j=1; j<10; j=j+2)
      {
        if (A[i] == B[j])
          {
            printf("C: %d, As: %d, Bs: %d.\n", A[i], A[i-1], B[j-1]);
            break;
           }
      }
  }
```

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