Problem 1 (2 parts, 30 points)

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.

```c
int j = ... ;  // given
int A[100] = {22,-41,10001,...42}; // given
int index = 0;
int i;
for (i = 0; i<100; i++)
{
    if (A[i] < A[j])
    {
        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.

<table>
<thead>
<tr>
<th>Label</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A:</td>
<td>.data</td>
<td># given set A</td>
</tr>
<tr>
<td></td>
<td>.word 22, -41, 10001, ..., 42</td>
<td># given set A</td>
</tr>
<tr>
<td></td>
<td>.text</td>
<td># initialize j</td>
</tr>
<tr>
<td></td>
<td>addi $1, $0, ...</td>
<td># initialize index</td>
</tr>
<tr>
<td></td>
<td>addi $2, $0, 0</td>
<td># initialize loop counter</td>
</tr>
<tr>
<td></td>
<td>addi $3, $0, 0</td>
<td># scale j by 4 to look up A[j]</td>
</tr>
<tr>
<td></td>
<td>sll $1, $1, 2</td>
<td># $5: A[j]</td>
</tr>
<tr>
<td></td>
<td>lw $5, A($1)</td>
<td># lookup current element of A</td>
</tr>
<tr>
<td>Loop:</td>
<td>slti $4, $3, 400</td>
<td># is loop counter &lt; limit?</td>
</tr>
<tr>
<td></td>
<td>beq $4, $0, Exit</td>
<td># if not, exit loop</td>
</tr>
<tr>
<td></td>
<td>lw $6, A($3)</td>
<td># lookup current element of A</td>
</tr>
</tbody>
</table>
|       | slt $4, $6, $5 | # is A[i] < A[j]?
|       | beq $4, $6, Skip | # if not, Skip increment |
|       | addi $2, $2, 1 | # increment index |
| Skip: | addi $3, $3, 4 | # increment loop counter |
|       | j Loop | # continue looping |
| Exit: | jr $31 | #
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.

\begin{verbatim}
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:
\end{verbatim}

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.

\begin{verbatim}
if (x || ((y>=10) && (z<w)))
    x = z+w;
else
    x = y+z;
OR
if (!x && ((y<10) || (z>=w)))
    x = y+z;
else
    x = z+w;
\end{verbatim}
Problem 3 (4 parts, 21 points)  

**MIPS Equivalences**

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

<table>
<thead>
<tr>
<th>Original:</th>
<th>Equivalent MIPS statement:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ori $3, $0, SetA</td>
<td>addi $4, $1, SetA</td>
</tr>
<tr>
<td>add $4, $3, $1</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Original:</th>
<th>Equivalent MIPS statement:</th>
</tr>
</thead>
<tbody>
<tr>
<td>lui $4, 0xFF00</td>
<td></td>
</tr>
<tr>
<td>lw $3, 1000($0)</td>
<td></td>
</tr>
<tr>
<td>and $3, $3, $4</td>
<td></td>
</tr>
<tr>
<td>srl $3, $3, 24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1bu $3, 1003($0)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Original:</th>
<th>Equivalent MIPS in two instructions only:</th>
</tr>
</thead>
<tbody>
<tr>
<td>slt $3, $2, $1</td>
<td>slt $3, $1, $2</td>
</tr>
<tr>
<td>bne $3, $0, Target</td>
<td>beq $3, $0, Target</td>
</tr>
<tr>
<td>beq $1, $2, Target</td>
<td></td>
</tr>
</tbody>
</table>

**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 $2$: 0xFFFFFFFFCD
Problem 4 (2 parts, 25 points)  
Nonlocal Control Flow

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

```c
int i, x;

for(i=0; i<10; i++){
    if(i & 1) \ bitise
        continue;
    x = A[i];
    printf("x = %d\n", x);
}
```

```
x = 99
x = 44
x = 56
x = 1
x = 9
```

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

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

for(__ i=0__; ___ i<10____; _ i += 2_ ){
    x = A[i];
    printf("x = %d\n", x);
}
```

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

```c
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? 4
- How many times is `continue` executed? 8
- How many `printf` statements are executed? 24