Q1) Convert the following C code to MIPS:

```
int globalValue = 10;

void main()
{
        globalValue = addToGlobal(5);
}

int addToGlobal(int num)
{
        if (num == globalValue)
                return num + 10;
        else
                return 0;
}
```

Answer to Q1;

```
.data
        globalValue: .word 10
.text
        main:
        addi $a0, $zero, 5              # 1st argument = 5
        jal addToGlobal                 # call function
        sw $v0, globalValue($zero)      # globalValue = v0  (returned value)
        addi $v0, $zero, 10             # v0 = 10 (to exit program)
        syscall                         # exit Program

        addToGlobal:
        lw $t0, globalValue($zero)      # t0 = globalValue
        bne $a0, $t0, ELSE              # if t0 != num goto ELSE
        addi $v0, $a0, 10          # return value = num + 10
        jr $ra                          # return to caller
        ELSE: addi $v0, $zero, 0        # return value = 0
        jr $ra                          # return to caller
```

Q2) Convert the following C code to MIPS:

```
int val1 = 10;
int val2 = 20;
int returnValue;

int returnMax(int num1, int num2)
{
        if (num1 >= num2)
                return num1;
        else
                return num2;
}

void main()
{
        returnValue = returnMax(val1, val2);
}
```

Answer to Q2;

```
.data
        val1: .word 10
        val2: .word 20
        returnValue: .space 4
.text
        main:
                lw $a0, val1($zero)             # 01. Load the content of val1 to a0
                lw $a1, val2($zero)             # 02. Load the content of val2 to a2
                jal returnMax                   # 03. Transfer control to 'returnMax'
                sw $v0, returnValue($zero)      # 08. Assign 'returnValue' v0
                addi $v0, $zero, 10             # 09. Quit program step 1
                syscall                         # 10. Quit program step 2
        returnMax:
                slt $t0, $a0, $a1               # 04. Evaluate boolean expression
                bne $t0, $zero, ELSE            # 05. Evaluate boolean expression
                add $v0, $a0, $zero             # 06a. Place a0 in v0; to be returned
                jr $ra                          # 07a. Go back to 'main'
                ELSE: add $v0, $a1, $zero       # 06b. Place a1 in v0; to be returned
                jr $ra                          # 07b. Go back to 'main'
```

Q3) Convert the following C code to MIPS:

```
void iniArray(int array[], int arraySize, int value)
{
        int i;
        for (i = 0; i < arraySize; i ++)
                array[i] = value;
}

void main()
{
        int myArray[100];
        iniArray(myArray, 100, 5);
}
```

```
.text
        main:
                addi $sp, $sp, -400      # Allocate space for the array
                addi $a0, $sp, 0         # a0 points to the array; 1st argument
                addi $a1, $zero, 100     # a1 is assigned 100; 2nd argument
                addi $a2, $zero, 5       # a2 is assigned 5; 3rd argument
                jal iniArray             # Transfer control to 'iniArray'
                addi $v0, $zero, 10
                syscall
        iniArray:
                addi $t0, $zero, 0               # int i = 0; needed for the loop
                mainLoop:
                slt $t1, $t0, $a1                # Evaluate boolean expression
                bne $t1, $zero, beginLoop        # if i < arraySize, go to 'beginLoop'
                jr $ra                           # Condition false; quit the function
                beginLoop:
                add $t2, $zero, $a0              # place base address of array in t2
                add $t2, $t2, $t0                # t2 = t2 + i  * We add i four
                add $t2, $t2, $t0                # t2 = t2 + i  * times to account for
                add $t2, $t2, $t0                # t2 = t2 + i  * the offset in bytes
                add $t2, $t2, $t0                # t2 = t2 + i  * Ex. on next line;
                # array[i] becomes address of 'array'+i+i+i+i (counting in bytes)
                sw $a2, 0($t2)                   # a[i] = a2
                addi $t0, $t0, 1                 # i = i + 1
                j mainLoop                       # Go to beginning of loop
```