```
#############################################################################
# Name:           Noah del Angel
# Class:          CS2318-002, Fall 2020
# Subject:        Optional Assignment 1
# Date:                          12/6/2020
#############################################################################
#int  hasDup(int a[], int n);
#int  exists(int a[], int n, int target);
#void CoutCstr(char cstr[]);
#void CoutCstrNL(char cstr[]);
#void CoutOneInt(int oneInt);
#void PopulateArray(int a[], int* usedPtr, int cap);
#void ShowArray(int a[], int used);
#void ShowArrayLabeled(int a[], int used, char label[]);
#
                .text
                .globl main
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#int main()
#############################################################################
main:
#################
# Register usage:
#################
# $t4: reply
# $v1: holder for a value/address
# (usual ones for syscall & function call)
#############################################################################
                # PROLOG:
                addiu $sp, $sp, -208
                sw $ra, 204($sp)
                sw $fp, 200($sp)
                addiu $fp, $sp, 208
#{
                # BODY:
#               int a1[12];
#               int used1;
#               char reply;
#               char ueStr[]     = "u-entered a1: ";
#               char dup0Msg[]   = "a1 has no dups";
#               char dupsMsg[]   = "a1 has 1+ dups";
#               char dacStr[]    = "Do another case? (n or N = no, others = yes) ";
#               char dlStr[]     = "================================";
#               char byeStr[]    = "bye...";
                j begDataInitM          # "clutter-reduction" jump
endDataInitM:
#               reply = 'y';
                li $t4, 'y'
#               goto WTest1;
                j WTest1
begW1:
#               PopulateArray(a1, &used1, 12);
                addi $a0, $sp, 152
                addi $a1, $sp, 148
                li $a2, 12
                jal PopulateArray
#               ShowArrayLabeled(a1, used1, ueStr);
                addi $a0, $sp, 152
                lw $a1, 148($sp)
                addi $a2, $sp, 16
                jal ShowArrayLabeled

#               ************************
#               ********** (9) **********
```

```
#                ************************
                 addi $a0, $sp, 152      # $a0 has a1
                 lw $a1, 148($sp)        # $a1 has used1
                 jal hasDup
#                if ( hasDup(a1, used1) != 0 )
                 beqz $v0, else1N
#                {
#                   CoutCstrNL(dupsMsg);
                 addi $a0, $sp, 31 # a0 has dupMsg
                 jal CoutCstrNL
                 j endif1

#                }
#                else
                 else1N:
#                {
#                   CoutCstrNL(dup0Msg);
                 addi $a0, $sp, 46 # $a0 has dup0Msg
                 jal CoutCstrNL

#                }
                 endif1:


#                cout << dacStr;
                 addi $a0, $sp, 94
                 jal CoutCstr
#                cin >> reply;
                 li $v0, 12
                 syscall
                 move $t4, $v0           # $t4 is reply
                 # newline to offset shortcoming of syscall #12
                 li $v0, 11
                 li $a0, '\n'
                 syscall
WTest1:
#                if (reply == 'n') goto xitW1;
#                if (reply != 'N') goto begW1;
                 li $v1, 'n'
                 beq $t4, $v1, xitW1
                 li $v1, 'N'
                 bne $t4, $v1, begW1
xitW1:
#                cout << dlStr << '\n';
                 addi $a0, $sp, 61
                 jal CoutCstrNL
#                cout << byeStr << '\n';
                 addi $a0, $sp, 140
                 jal CoutCstrNL
#                cout << dlStr << '\n';
                 addi $a0, $sp, 61
                 jal CoutCstrNL
#                return 0;
#}
                 # EPILOG:
                 lw $fp, 200($sp)
                 lw $ra, 204($sp)
                 addiu $sp, $sp, 208
                 li $v0, 10
                 syscall


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#int  hasDup(int* arrBegPtr, int numEle)
```

```
################################################################################
hasDup:
#################
# Register usage:
#################
# ...
# $v1: holder for a value/address
################################################################################
#                ************************
#                ********** (27) **********
#                ************************
                 # PROLOG:
                 addiu $sp, $sp, -32     # create stack
                 sw $ra, 28($sp)
                 sw $fp, 24($sp)
                 addiu $fp, $sp, 32      # store new frame pointer
                 sw $a3, 12($fp)         # store $a3 placeholder
                 sw $a2, 8($fp)          # store $a2 placeholder
                 sw $a1, 4($fp)          # 4($sp) has second param
                 sw $a0, 0($fp)          # 0($sp) has first param
#{
                 # BODY:

                 lw $t1, 0($sp)          # $t1 has arrBegPtr
                 lw $t2, 4($sp)          # $t2 has numEle
#                if (numEle <= 1)
                 li $t3, 1               # $t3 has 1
                 bge $t2, $t3, else2N
#                {
#                   return 0;
                    li $v0, 0
                    j hasDupEpilog
#                }
                 else2N:

                 # call exists

                 addi $a0, $a0, 4        # a0 has arrBegPtr + 1
                 addi $a1, $a1, -1       # a1 has numEle - 1
                 lw $a2, 0($fp)          # a2 has arrBegPtr
                 jal exists

                 lw $a0, 0($fp)          # reload pointer


#                if ( exists(arrBegPtr + 1, numEle - 1, *arrBegPtr) != 0 )
                 beqz $v0, else3N
#                {
#                   return 1;
                    li $v0, 1
                    j hasDupEpilog
#                }
                 else3N:

                 # call hasDup
                 addi $a0, $a0, 4
                 lw $a1, 4($fp)
                 addi $a1, $a1, -1
                 jal hasDup
#                return hasDup(arrBegPtr + 1, numEle -1);
#}
                 # EPILOG:
                 hasDupEpilog:
                 lw $fp, 24($sp)
                 lw $ra, 28($sp)
```

```
                addiu $sp, $sp, 32
                jr $ra
########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
                li $t9, 999999999
                #li $v0, 999999999      # don't clobber actual return value
                li $v1, 999999999
########################################
                jr $ra


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#int  exists(int* arrBegPtr, int numEle, int target)
################################################################################
exists:
#################
# Register usage:
#################
# $v1: holder for a value/address
################################################################################
#                       *************************
#                       ********** (17) **********
#                       *************************
                # PROLOG:
                addiu $sp, $sp, -32
                sw $ra, 28($sp)
                sw $fp, 24($sp)
                addiu $fp, $sp, 32      # set new frame pointer
                sw $a3, 12($fp)
                sw $a2, 8($fp)          # 8($fp) has the third param
                sw $a1, 4($fp)          # 4($fp) has the second param
                sw $a0, 0($fp)          # 0($fp) has the first param
#{
                # BODY:
                lw $t0, 0($a0)          # $t0 has arrBegPtr
                lw $t2, 0($a2)          # $t2 has target
#               if (numEle <= 0)
                bgtz $a1, else4N
#               {
#                  return 0;
                   li $v0, 0
                   j existsEpilog
#               }
                else4N:

#               if (*arrBegPtr == target)
                bne $t0, $t2, else5N
```

```
#                   {
#                      return 1;
                       li $v0, 1
                       j existsEpilog
#                   }
                    else5N:
                    addi $a0, $a0, 4
                    addi $a1, $a1, -1


#                   return exists(arrBegPtr + 1, numEle - 1, target);
                    jal exists
#}
                    # EPILOG:
                    existsEpilog:
                    lw $ra, 28($sp)
                    lw $fp, 24($sp)
                    addiu $sp, $sp, 32
                    jr $ra

########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                    li $a0, 999999999
                    li $a1, 999999999
                    li $a2, 999999999
                    li $a3, 999999999
                    li $t0, 999999999
                    li $t1, 999999999
                    li $t2, 999999999
                    li $t3, 999999999
                    li $t4, 999999999
                    li $t5, 999999999
                    li $t6, 999999999
                    li $t7, 999999999
                    li $t8, 999999999
                    li $t9, 999999999
                    #li $v0, 999999999      # don't clobber actual return value
                    li $v1, 999999999
########################################
                    jr $ra


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#void CoutCstr(char cstr[])
################################################################################
CoutCstr:
#################
# Register usage:
#################
# (usual ones for syscall)
################################################################################
                    # PROLOG:
                    # no stack frame needed
#{
                    # BODY:
#                   cout << cstr;
                    li $v0, 4
                    syscall
#}
                    # EPILOG:
########################################
```

```
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
                li $t9, 999999999
                li $v0, 999999999
                li $v1, 999999999
########################################
                jr $ra


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#void CoutCstrNL(char cstr[])
##############################################################################
CoutCstrNL:
#################
# Register usage:
#################
# (usual ones for syscall & function call)
##############################################################################
                # PROLOG:
                addiu $sp, $sp, -32
                sw $ra, 28($sp)
                sw $fp, 24($sp)
                addiu $fp, $sp, 32
#{
                # BODY:
#               CoutCstr(cstr);
                jal CoutCstr
#               cout << '\n';
                li $a0, '\n'
                li $v0, 11
                syscall
#}
                # EPILOG:
                lw $fp, 24($sp)
                lw $ra, 28($sp)
                addiu $sp, $sp, 32
########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
```

```
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
                li $t9, 999999999
                li $v0, 999999999
                li $v1, 999999999
########################################
                jr $ra


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#void CoutOneInt(int oneInt)
##########################################################################################
CoutOneInt:
#################
# Register usage:
#################
# (usual ones for syscall)
##########################################################################################
                # PROLOG:
                # no stack frame needed
#{
                # BODY:
#               cout << oneInt;
                li $v0, 1
                syscall
#}
                # EPILOG:
########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
                li $t9, 999999999
                li $v0, 999999999
                li $v1, 999999999
########################################
                jr $ra


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#void PopulateArray(int a[], int* usedPtr, int cap)
##########################################################################################
PopulateArray:
#################
# Register usage:
#################
# $s1: hopPtr
```

```
# $t0: holder for a value/address
# $t1: another holder for a value/address
# $t2: yet another holder for a value/address
# $t4: reply
# (usual ones for syscall & function call)
#################################################################################
                # PROLOG:
                addiu $sp, $sp, -120
                sw $ra, 116($sp)
                sw $fp, 112($sp)
                addiu $fp, $sp, 120

                sw $a1, 4($fp)          # usedPtr as received saved in caller's frame
                sw $a2, 8($fp)          # cap as received saved in caller's frame
                sw $s1, 16($sp)         # save $s1 (callee-saved)
#{
                # BODY:
#               char reply;
#               char einStr[]    = "Enter integer #";
#               char moStr[]     = "Max of ";
#               char ieStr[]     = " ints entered...";
#               char emiStr[]    = "Enter more ints? (n or N = no, others = yes) ";
#               int *hopPtr;
                j begDataInitPA         # "clutter-reduction" jump
endDataInitPA:
#               reply = 'y';
                li $t4, 'y'             # $t4 is reply
#               *usedPtr = 0;
                sw $0, 0($a1)           # $a1 still has usedPtr as received
#               hopPtr = a;
                move $s1, $a0           # $a0 still has a as received
#               goto WTest2;
                j WTest2
begW2:
#               CoutCstr(einStr);
                addi $a0, $sp, 24
                jal CoutCstr
#               CoutOneInt(*usedPtr + 1);
                lw $a1, 4($fp)          # usedPtr as received re-loaded into $a1
                # CoutCstr might have clobbered $a1
                lw $a0, 0($a1)          # $a0 has *usedPtr
                addi $a0, $a0, 1        # *usedPtr + 1 as arg1
                jal CoutOneInt
#               cout << ':' << ' ';
                li $v0, 11
                li $a0, ':'
                syscall
                li $a0, ' '
                syscall
#               cin >> *hopPtr;
                li $v0, 5
                syscall                 # $v0 has user-entered int
                sw $v0, 0($s1)          # $s1 is hopPtr
#               ++(*usedPtr);
                lw $a1, 4($fp)          # usedPtr as received re-loaded into $a1
                # CoutOneInt might have clobbered $a1
                lw $t1, 0($a1)          # $t1 has *usedPtr
                addi $t1, $t1, 1        # $t1 has *usedPtr + 1
                sw $t1, 0($a1)          # ++(*usedPtr)
#               ++hopPtr;
                addi $s1, $s1, 4        # $s1 is hopPtr
#               if (*usedPtr >= cap) goto else1;
                lw $a2, 8($fp)          # cap as received re-loaded into $a2
                # CoutOneInt might have clobbered $a2
                bge $t1, $a2, else1     # if (*usedPtr >= cap) goto else1
```

```
                # $t1 still has up-to-date *usedPtr
#               CoutCstr(emiStr);
                addi $a0, $sp, 48
                jal CoutCstr
#               cin >> reply;
                li $v0, 12
                syscall
                move $t4, $v0           # $t4 is reply
                # newline to offset shortcoming of syscall #12
                li $v0, 11
                li $a0, '\n'
                syscall
#               goto endI1;
                j endI1
else1:
#               CoutCstr(moStr);
                addi $a0, $sp, 40
                jal CoutCstr
#               CoutOneInt(cap);
                lw $a0, 8($fp)          # cap as received loaded into $a0
                # not using $a2 as CoutCstr might have clobbered it
                jal CoutOneInt
#               CoutCstr(ieStr);
                addi $a0, $sp, 94
                jal CoutCstr
#               cout << endl;
                li $v0, 11
                li $a0, '\n'
                syscall
#               reply = 'n';
                li $t4, 'n'             # $t4 is reply
endI1:
WTest2:
#               if (reply == 'n') goto xitW2;
#               if (reply != 'N') goto begW2;
                li $t0, 'n'
                beq $t4, $t0, xitW2
                li $t0, 'N'
                bne $t4, $t0, begW2
xitW2:
#               return;
#}
                # EPILOG:
                lw $s1, 16($sp)         # restore $s1 (callee-saved)
                lw $fp, 112($sp)
                lw $ra, 116($sp)
                addiu $sp, $sp, 120
########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
```

```
                li $t9, 999999999
                li $v0, 999999999
                li $v1, 999999999
########################################
                jr $ra


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#void ShowArray(int a[], int used)
#{
################################################################################
ShowArray:
#################
# Register usage:
#################
# $t1: hopPtr
# $t9: endPtr
# $a1: used (as received)
# (usual ones for syscall & function call)
################################################################################
                # PROLOG:
                # no stack frame needed
                # BODY:
#               int *hopPtr;
#               int *endPtr;

#               if (used <= 0) goto endI2;
                blez $a1, endI2
#               hopPtr = a;
                move $t1, $a0
#               endPtr = a + used;
                move $t9, $a1           # $t9 has used
                sll $t9, $t9, 2         # $t9 has 4*used
                add $t9, $t9, $t1       # $t9 has &a[used]
begDW1:
#               cout << *hopPtr << ' ' << ' ';
                li $v0, 1
                lw $a0, 0($t1)          # $a0 has *hopPtr
                syscall
                li $v0, 11
                li $a0, ' '
                syscall
                syscall
#               ++hopPtr;
                addi $t1, $t1, 4
endDW1:
DWTest1:
#               if (hopPtr < endPtr) goto begDW1;
                blt $t1, $t9, begDW1
endI2:
#               cout << endl;
                li $v0, 11
                li $a0, '\n'
                syscall
#}
                # EPILOG:
########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
```

```
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
                li $t9, 999999999
                li $v0, 999999999
                li $v1, 999999999
########################################
                jr $ra


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#void ShowArrayLabeled(int a[], int used, char label[])
##############################################################################
ShowArrayLabeled:
#################
# Register usage:
#################
# $t1: i
# $v1: holder for a value/address
# (usual ones for function call)
##############################################################################
                # PROLOG:
                addiu $sp, $sp, -32
                sw $ra, 28($sp)
                sw $fp, 24($sp)
                addiu $fp, $sp, 32

                sw $a0, 0($fp)          # a as received saved in caller's frame
                sw $a1, 4($fp)          # used as received saved in caller's frame
#{
                # BODY:
#               CoutCstr(label);
                move $a0, $a2
                jal CoutCstr
#               ShowArray(a, used);
                lw $a0, 0($fp)          # a as received re-loaded into $a0
                lw $a1, 4($fp)          # used as received re-loaded into $a1
                # CoutCstr might have clobbered $a0 & $a1
                jal ShowArray
#}
                # EPILOG:
                lw $fp, 24($sp)
                lw $ra, 28($sp)
                addiu $sp, $sp, 32
########################################
# deliberate clobbering of caller-saved
# (meant to catch improper presumptions -
# no effect if no such presumptions made)
########################################
                li $a0, 999999999
                li $a1, 999999999
                li $a2, 999999999
                li $a3, 999999999
                li $t0, 999999999
                li $t1, 999999999
                li $t2, 999999999
                li $t3, 999999999
```

```
                li $t4, 999999999
                li $t5, 999999999
                li $t6, 999999999
                li $t7, 999999999
                li $t8, 999999999
                li $t9, 999999999
                li $v0, 999999999
                li $v1, 999999999
########################################
                jr $ra


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
# main's string initialization code moved out of the way to reduce clutter
################################################################################
begDataInitM:
                li $t0, 'u'
                sb $t0, 16($sp)
                li $t0, '-'
                sb $t0, 17($sp)
                li $t0, 'e'
                sb $t0, 18($sp)
                li $t0, 'n'
                sb $t0, 19($sp)
                li $t0, 't'
                sb $t0, 20($sp)
                li $t0, 'e'
                sb $t0, 21($sp)
                li $t0, 'r'
                sb $t0, 22($sp)
                li $t0, 'e'
                sb $t0, 23($sp)
                li $t0, 'd'
                sb $t0, 24($sp)
                li $t0, ' '
                sb $t0, 25($sp)
                li $t0, 'a'
                sb $t0, 26($sp)
                li $t0, '1'
                sb $t0, 27($sp)
                li $t0, ':'
                sb $t0, 28($sp)
                li $t0, ' '
                sb $t0, 29($sp)
                li $t0, '\0'
                sb $t0, 30($sp)
                li $t0, 'a'
                sb $t0, 31($sp)
                li $t0, '1'
                sb $t0, 32($sp)
                li $t0, ' '
                sb $t0, 33($sp)
                li $t0, 'h'
                sb $t0, 34($sp)
                li $t0, 'a'
                sb $t0, 35($sp)
                li $t0, 's'
                sb $t0, 36($sp)
                li $t0, ' '
                sb $t0, 37($sp)
                li $t0, '1'
                sb $t0, 38($sp)
                li $t0, '+'
                sb $t0, 39($sp)
```

```
li $t0, ' '
sb $t0, 40($sp)
li $t0, 'd'
sb $t0, 41($sp)
li $t0, 'u'
sb $t0, 42($sp)
li $t0, 'p'
sb $t0, 43($sp)
li $t0, 's'
sb $t0, 44($sp)
li $t0, '\0'
sb $t0, 45($sp)
li $t0, 'a'
sb $t0, 46($sp)
li $t0, '1'
sb $t0, 47($sp)
li $t0, ' '
sb $t0, 48($sp)
li $t0, 'h'
sb $t0, 49($sp)
li $t0, 'a'
sb $t0, 50($sp)
li $t0, 's'
sb $t0, 51($sp)
li $t0, ' '
sb $t0, 52($sp)
li $t0, 'n'
sb $t0, 53($sp)
li $t0, 'o'
sb $t0, 54($sp)
li $t0, ' '
sb $t0, 55($sp)
li $t0, 'd'
sb $t0, 56($sp)
li $t0, 'u'
sb $t0, 57($sp)
li $t0, 'p'
sb $t0, 58($sp)
li $t0, 's'
sb $t0, 59($sp)
li $t0, '\0'
sb $t0, 60($sp)
li $t0, '='
sb $t0, 61($sp)
li $t0, '='
sb $t0, 62($sp)
li $t0, '='
sb $t0, 63($sp)
li $t0, '='
sb $t0, 64($sp)
li $t0, '='
sb $t0, 65($sp)
li $t0, '='
sb $t0, 66($sp)
li $t0, '='
sb $t0, 67($sp)
li $t0, '='
sb $t0, 68($sp)
li $t0, '='
sb $t0, 69($sp)
li $t0, '='
sb $t0, 70($sp)
li $t0, '='
sb $t0, 71($sp)
li $t0, '='
```

```
sb $t0, 72($sp)
li $t0, '='
sb $t0, 73($sp)
li $t0, '='
sb $t0, 74($sp)
li $t0, '='
sb $t0, 75($sp)
li $t0, '='
sb $t0, 76($sp)
li $t0, '='
sb $t0, 77($sp)
li $t0, '='
sb $t0, 78($sp)
li $t0, '='
sb $t0, 79($sp)
li $t0, '='
sb $t0, 80($sp)
li $t0, '='
sb $t0, 81($sp)
li $t0, '='
sb $t0, 82($sp)
li $t0, '='
sb $t0, 83($sp)
li $t0, '='
sb $t0, 84($sp)
li $t0, '='
sb $t0, 85($sp)
li $t0, '='
sb $t0, 86($sp)
li $t0, '='
sb $t0, 87($sp)
li $t0, '='
sb $t0, 88($sp)
li $t0, '='
sb $t0, 89($sp)
li $t0, '='
sb $t0, 90($sp)
li $t0, '='
sb $t0, 91($sp)
li $t0, '='
sb $t0, 92($sp)
li $t0, '\0'
sb $t0, 93($sp)
li $t0, 'D'
sb $t0, 94($sp)
li $t0, 'o'
sb $t0, 95($sp)
li $t0, ' '
sb $t0, 96($sp)
li $t0, 'a'
sb $t0, 97($sp)
li $t0, 'n'
sb $t0, 98($sp)
li $t0, 'o'
sb $t0, 99($sp)
li $t0, 't'
sb $t0, 100($sp)
li $t0, 'h'
sb $t0, 101($sp)
li $t0, 'e'
sb $t0, 102($sp)
li $t0, 'r'
sb $t0, 103($sp)
li $t0, ' '
sb $t0, 104($sp)
```

```
li $t0, 'c'
sb $t0, 105($sp)
li $t0, 'a'
sb $t0, 106($sp)
li $t0, 's'
sb $t0, 107($sp)
li $t0, 'e'
sb $t0, 108($sp)
li $t0, '?'
sb $t0, 109($sp)
li $t0, ' '
sb $t0, 110($sp)
li $t0, '('
sb $t0, 111($sp)
li $t0, 'n'
sb $t0, 112($sp)
li $t0, ' '
sb $t0, 113($sp)
li $t0, 'o'
sb $t0, 114($sp)
li $t0, 'r'
sb $t0, 115($sp)
li $t0, ' '
sb $t0, 116($sp)
li $t0, 'N'
sb $t0, 117($sp)
li $t0, ' '
sb $t0, 118($sp)
li $t0, '='
sb $t0, 119($sp)
li $t0, ' '
sb $t0, 120($sp)
li $t0, 'n'
sb $t0, 121($sp)
li $t0, 'o'
sb $t0, 122($sp)
li $t0, ','
sb $t0, 123($sp)
li $t0, ' '
sb $t0, 124($sp)
li $t0, 'o'
sb $t0, 125($sp)
li $t0, 't'
sb $t0, 126($sp)
li $t0, 'h'
sb $t0, 127($sp)
li $t0, 'e'
sb $t0, 128($sp)
li $t0, 'r'
sb $t0, 129($sp)
li $t0, 's'
sb $t0, 130($sp)
li $t0, ' '
sb $t0, 131($sp)
li $t0, '='
sb $t0, 132($sp)
li $t0, ' '
sb $t0, 133($sp)
li $t0, 'y'
sb $t0, 134($sp)
li $t0, 'e'
sb $t0, 135($sp)
li $t0, 's'
sb $t0, 136($sp)
li $t0, ')'
```

```
                sb $t0, 137($sp)
                li $t0, ' '
                sb $t0, 138($sp)
                li $t0, '\0'
                sb $t0, 139($sp)
                li $t0, 'b'
                sb $t0, 140($sp)
                li $t0, 'y'
                sb $t0, 141($sp)
                li $t0, 'e'
                sb $t0, 142($sp)
                li $t0, '.'
                sb $t0, 143($sp)
                li $t0, '.'
                sb $t0, 144($sp)
                li $t0, '.'
                sb $t0, 145($sp)
                li $t0, '\0'
                sb $t0, 146($sp)
                j endDataInitM          # back to main

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
# PopulateArray's string initialization code moved out of the way to reduce clutter
################################################################################
begDataInitPA:
                li $t0, 'E'
                sb $t0, 24($sp)
                li $t0, 'n'
                sb $t0, 25($sp)
                li $t0, 't'
                sb $t0, 26($sp)
                li $t0, 'e'
                sb $t0, 27($sp)
                li $t0, 'r'
                sb $t0, 28($sp)
                li $t0, ' '
                sb $t0, 29($sp)
                li $t0, 'i'
                sb $t0, 30($sp)
                li $t0, 'n'
                sb $t0, 31($sp)
                li $t0, 't'
                sb $t0, 32($sp)
                li $t0, 'e'
                sb $t0, 33($sp)
                li $t0, 'g'
                sb $t0, 34($sp)
                li $t0, 'e'
                sb $t0, 35($sp)
                li $t0, 'r'
                sb $t0, 36($sp)
                li $t0, ' '
                sb $t0, 37($sp)
                li $t0, '#'
                sb $t0, 38($sp)
                li $t0, '\0'
                sb $t0, 39($sp)
                li $t0, 'M'
                sb $t0, 40($sp)
                li $t0, 'a'
                sb $t0, 41($sp)
                li $t0, 'x'
                sb $t0, 42($sp)
                li $t0, ' '
```

```
sb $t0, 43($sp)
li $t0, 'o'
sb $t0, 44($sp)
li $t0, 'f'
sb $t0, 45($sp)
li $t0, ' '
sb $t0, 46($sp)
li $t0, '\0'
sb $t0, 47($sp)
li $t0, 'E'
sb $t0, 48($sp)
li $t0, 'n'
sb $t0, 49($sp)
li $t0, 't'
sb $t0, 50($sp)
li $t0, 'e'
sb $t0, 51($sp)
li $t0, 'r'
sb $t0, 52($sp)
li $t0, ' '
sb $t0, 53($sp)
li $t0, 'm'
sb $t0, 54($sp)
li $t0, 'o'
sb $t0, 55($sp)
li $t0, 'r'
sb $t0, 56($sp)
li $t0, 'e'
sb $t0, 57($sp)
li $t0, ' '
sb $t0, 58($sp)
li $t0, 'i'
sb $t0, 59($sp)
li $t0, 'n'
sb $t0, 60($sp)
li $t0, 't'
sb $t0, 61($sp)
li $t0, 's'
sb $t0, 62($sp)
li $t0, '?'
sb $t0, 63($sp)
li $t0, ' '
sb $t0, 64($sp)
li $t0, '('
sb $t0, 65($sp)
li $t0, 'n'
sb $t0, 66($sp)
li $t0, ' '
sb $t0, 67($sp)
li $t0, 'o'
sb $t0, 68($sp)
li $t0, 'r'
sb $t0, 69($sp)
li $t0, ' '
sb $t0, 70($sp)
li $t0, 'N'
sb $t0, 71($sp)
li $t0, ' '
sb $t0, 72($sp)
li $t0, '='
sb $t0, 73($sp)
li $t0, ' '
sb $t0, 74($sp)
li $t0, 'n'
sb $t0, 75($sp)
```

```
li $t0, 'o'
sb $t0, 76($sp)
li $t0, ','
sb $t0, 77($sp)
li $t0, ' '
sb $t0, 78($sp)
li $t0, 'o'
sb $t0, 79($sp)
li $t0, 't'
sb $t0, 80($sp)
li $t0, 'h'
sb $t0, 81($sp)
li $t0, 'e'
sb $t0, 82($sp)
li $t0, 'r'
sb $t0, 83($sp)
li $t0, 's'
sb $t0, 84($sp)
li $t0, ' '
sb $t0, 85($sp)
li $t0, '='
sb $t0, 86($sp)
li $t0, ' '
sb $t0, 87($sp)
li $t0, 'y'
sb $t0, 88($sp)
li $t0, 'e'
sb $t0, 89($sp)
li $t0, 's'
sb $t0, 90($sp)
li $t0, ')'
sb $t0, 91($sp)
li $t0, ' '
sb $t0, 92($sp)
li $t0, '\0'
sb $t0, 93($sp)
li $t0, ' '
sb $t0, 94($sp)
li $t0, 'i'
sb $t0, 95($sp)
li $t0, 'n'
sb $t0, 96($sp)
li $t0, 't'
sb $t0, 97($sp)
li $t0, 's'
sb $t0, 98($sp)
li $t0, ' '
sb $t0, 99($sp)
li $t0, 'e'
sb $t0, 100($sp)
li $t0, 'n'
sb $t0, 101($sp)
li $t0, 't'
sb $t0, 102($sp)
li $t0, 'e'
sb $t0, 103($sp)
li $t0, 'r'
sb $t0, 104($sp)
li $t0, 'e'
sb $t0, 105($sp)
li $t0, 'd'
sb $t0, 106($sp)
li $t0, '.'
sb $t0, 107($sp)
li $t0, '.'
```

```
sb $t0, 108($sp)
li $t0, '.'
sb $t0, 109($sp)
li $t0, '\0'
sb $t0, 110($sp)
j endDataInitPA         # back to PopulateArray
```