```
1.
                                                 C code for problem 5
a) N
                                                 #include <stdio.h>
b) Y
                                                 #include <stdlib.h>
c) Y
                                                 #define SIZE 6
d) Y
                                                 #define MAX FIRST 10
                                                 int dud1 (void)
2. 1200 bytes
                                                   int i;
                                                   int x;
3.
                                                   int a[SIZE];
a) 171
                                                   int b[SIZE+SIZE];
b) -85
                                                   for (i=0; i<SIZE; i++)
c) (-1)*2^-2*1.375 = -0.34375
                                                        scanf("%d", &a[i]);
                                                        b[i] = 0;
4.
                                                       b[i+SIZE]=0;
a) Y
                                                     }
b) 0x100270
                                                   if (a[0]>MAX_FIRST)
c) grape
                                                     return 0;
                                                   x=a[0];
                                                   for (i=1;i<SIZE; i++)
5.
                                                     switch (a[i]) {
a.) 0x400575
                                                      case 2:
c.) 0x40057F
                                                        x=x-5;
d.) 0x4005CF
                                                      case 4:
e.) 0x4005C0
                                                        if (b[a[i]] == 1)
d.) answers may vary (see code)
                                                         return 0;
                                                       b[a[i]]=1;
                                                        x=x*4;
                                                        break;
                                                      case 3:
                                                        x*=2;
                                                        if (b[a[i]] == 1)
                                                         return 0;
                                                        b[a[i]]=1;
                                                        break;
                                                      case 6:
                                                        x+=7;
                                                        if (b[a[i]] == 1)
                                                         return 0;
                                                       b[a[i]]=1;
                                                       break;
                                                      case 7:
                                                        x+=3;
                                                        if (b[a[i]] == 1)
                                                         return 0;
                                                        b[a[i]]=1;
                                                        break;
                                                      case 8:
                                                        x = 17;
                                                        if (b[a[i]] == 1)
                                                         return 0;
                                                        b[a[i]]=1;
                                                      default:
                                                        x=x-1;
                                                   return (x==42);
```

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO

SANTA BARBARA • SANTA CRUZ

CS 33 Midterm

All answers must be written on the answer sheet (last page of the exam).

All work should be written directly on the exam, use the backs of pages if needed.

This is an open book, open notes quiz – but you cannot share books or notes. An ASCII table is on the second to last page if you need it.

I will follow the guidelines of the university in reporting academic misconduct – please do not cheat.

	NAME:					
	ID:					
Pr	oblem 1:					
Pr	Problem 2:					
Pr	Problem 3:					
Pr	Problem 4:					
Pr	Problem 5:					
Тс	Γotal:					

1. C If You Can Solve This (12 points): The following problem assumes the following declarations:

```
int x = rand();
int y = rand();
int z = rand();
float f= foo(); // f is not NaN
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
```

For the following C expressions, circle either Y or N (but not both).

Always True?

a. (0+f)-f == 0Y

N

b. (ux-uy) == (x-y)Y

N

c. $((((x<<1)^x)-x)>>1) == x \Rightarrow !((x&12) == 12)$ Y

N

d. $(x^y)^z == y \Rightarrow x==z$ Y

N

Note that " \Rightarrow " represents an *implication*. A \Rightarrow B means that you assume A is true, and your answer should indicate whether B should be implied by A – i.e. given that A is true, is B always true?

2. What Size Is It? (4 points): Consider the following structure definition:

```
struct product {
    unsigned short ID;
    union NameOrSerial {
        char name[8];
        unsigned int serial;
    } label;
} my_data[100];
```

How many bytes would my_data consume in memory on an x86-64 Linux machine?

3.	Four	Perspectives	on a Bit	Vector (9 points):	Consider the following 8 b	oits:
----	------	--------------	----------	----------	------------	----------------------------	-------

10101011

We will interpret these bits in three different ways (assume the above is in big endian form):

- a. An 8-bit unsigned integer
- b. An 8-bit two's complement integer
- c. The 8-bit floating point format we covered in class

4. This Problem is a Pain in My Big Endian (35 points): Consider the following structure declaration:

```
struct my_struct{
  int val;
  char a;
  char b;
  short small_num;
  char * label;
  struct my_struct * left_child;
  struct my_struct * right_child;
};
```

We compile code using this struct on a 32-bit little endian machine, and insert a number of nodes into the structure. The following gdb interaction provides enough details to reverse engineer a tree built from these structs.

```
(qdb) print the_top
$3 = (struct my\_struct *) 0x100130
(qdb) x/96x 0x100130
0x100130:
           0x0000000a
                       0x00d91107
                                  0x00001f8e
                                               0x001001f0
0x100140:
           0x00100210
                       0x00000000 0x00000000
                                               0x00000000
0x100150:
           0x00000005
                       0x00c8020a 0x00001f71
                                               0x00100190
0x100160:
           0x00100170
                       0x00000000 0x00000000
                                               0x00000000
<del>0x100170:</del> 0x00000012
                       0x00431e18 0x00001f76
                                              0x001001b0
0x100180: 0x00000000
                       0x00000000 0x00000000
                                               0x00000000
0x100190: 0x00000003
                       0x0055180d 0x00001f7a
                                               0x00000000
0x1001a0:
           0x00000000
                       0x00000000 0x00000000
                                               0x00000000
0x1001b0:
           0x000000d
                       0x00b3020c 0x00001f80
                                               0x001001d0
0x1001c0:
           0x00000000
                       0x00000000
                                  0x00000000
                                               0x00000000
0x1001d0: 0x00000008
                       0x00111707 0x00001f87
                                               0x00000000
0x1001e0: 0x00000000 0x00000000 0x00000000
                                               0x00000000
                       0x002f1418 0x00001f95
0x1001f0: 0x00000005
                                               0x00100230
0x100200:
         0x00100250
                       0x00000000 0x00000000
                                               0x00000000
0x100210:
         0x0000000f
                       0x00050d11 0x00001f9a
                                              0x00100270
0x100220:
           0x00100290
                       0x00000000 0x00000000
                                               0x00000000
0x100230:
           0x00000003
                       0x006b1518 0x00001f9f
                                               0x00000000
         0x00000000
0x100240:
                       0x00000000 0x00000000
                                               0x00000000
         0x00000008
0x100250:
                       0x00070816 0x00001fa5
                                               0x00000000
                       0x00000000 0x00000000
0x100260:
           0x00000000
                                               0x00000000
0x100270: 0x0000000d 0x00481219
                                               0x00000000
           0x00000000 0x00000000 0x00000000
0x100280:
                                               0x00000000
0x100290:
           0x00000012
                                  0x00001fb1
                       0x00410213
                                               0x00000000
0x1002a0:
           0x00000000 0x00000000 0x00000000
                                               0x00000000
```

```
(gdb) x/32x 0x00001f60
0x1f60:
           0x00000000
                       0x00201868 0x1425ff00
                                               0x90000020
0x1f70:
           0x756c6200
                       0x65720065
                                  0x72670064
                                               0x006e6565
0x1f80:
           0x6c6c6579
                       0x7000776f 0x6c707275
                                               0x726f0065
0x1f90:
           0x65676e61
                       0x756c7000 0x6570006d
                                               0x6d007261
0x1fa0:
           0x6e6f6c65
                       0x72656200 0x67007972
                                               0x65706172
0x1fb0:
           0x70706100
                       0x0100656c 0x1c000000
                                               0x00000000
0x1fc0:
           0x1c000000 0x00000000 0x1c000000
                                               0x02000000
0x1fd0:
           0x90000000
                      0x3400000b 0x34000000
                                               0x31000000
```

Using this information, find the struct in the tree where val is equal to 13, and report the corresponding label (i.e. the actual string). Also, please fill in the blanks we have on the answer key with the requested intermediate values that would help you answer this question.

Hint – don't forget that gdb reverses byte ordering within each 4-byte chunk. So in the following dump:

```
(gdb) x/4x 0x00111110
0x111110: 0x33221100 0x77665544 0xBBAA9988 0xFFEEDDCC
```

This prints out 16 bytes of memory starting at address 0x111110. In this example, the 16 bytes of memory starting at 0x111110 would contain, in order from lowest address (0x111110) to highest address (0x11111F): 00112233445566778899AABBCCDDEEFF

So address 0x111110 contains the byte 0x00, address 0x111111 contains the byte 0x11, address 0x1111112 contains the byte 0x22, and so on. So in terms of just the least significant hex place of the address, gdb is actually printing out addresses in the following order:

3210 7654 BA98 FEDC

This is useful when reading words, but can be confusing for other values.

5. **Revenge of the Bomb Lab (40 points):** Here's your chance to show your bomb lab skills. Below we show the disassembled function dud1() – compiled on an x86-64 machine. The function does not take any parameters – but does read some number of integers, one at a time, using scanf (this is the *callq* instruction below). Your job is to provide the input that will result in this function returning the value 1 (i.e. 0x1 will be in %rax).

To help you with this task – we provide part of the C code for the function below (note that MAX_FIRST and SIZE are macros that would be replaced with constants at compile time):

```
int dud1(void)
  int i;
  int x;
  int a[SIZE];
  int b[SIZE+SIZE];
  for (i=0; i<SIZE; i++)
      scanf("%d", &a[i]);
      b[i] = 0;
      b[i+SIZE]=0;
    }
  if (a[0]>MAX_FIRST)
    return 0;
  x=a[0];
  for (...)
    switch (...) {
    ... /* some number of cases go here */
    }
 return ...;
}
```

where the ...'s above represent missing code you need to figure out.

In addition to the input string that would result in this function returning a 1, please fill in the blanks we have on the answer key with the requested intermediate values that would help you answer this question.

The next two pages contain everything you need for this problem.

Here's the function from objdump:

```
0000000000400540 <dud1>:
  400540:
                 41 54
                                            push
                                                    %r12
  400542:
                 55
                                                    %rbp
                                            push
  400543:
                 53
                                                    %rbx
                                            push
  400544:
                 31 db
                                                    %ebx, %ebx
                                            xor
                 48 83 ec 50
  400546:
                                            sub
                                                    $0x50,%rsp
                 4c 8d 64 24 30
                                                    0x30(%rsp),%r12
  40054a:
                                            lea
                 48 89 e5
  40054f:
                                            mov
                                                    %rsp,%rbp
  400552:
                 49 8d 34 1c
                                                    (%r12,%rbx,1),%rsi
                                            lea
  400556:
                 31 c0
                                                    %eax, %eax
                                            xor
                 bf 48 0b 40 00
                                                    $0x400b48, %edi
  400558:
                                            mov
  40055d:
                 e8 c6 fe ff ff
                                            callq
                                                    400428 <scanf@plt>
  400562:
                 c7 04 2b 00 00 00 00
                                                    $0x0,(%rbx,%rbp,1)
                                            movl
                 c7 44 2b 18 00 00 00
  400569:
                                            movl
                                                    $0x0,0x18(%rbx,%rbp,1)
                 00
  400570:
  400571:
                 48 83 c3 04
                                            add
                                                    $0x4,%rbx
                 48 83 fb 18
  400575:
                                                    $0x18,%rbx
                                            cmp
                 75 d7
  400579:
                                                    400552 <dud1+0x12>
                                            <del>jne</del>
  40057b:
                 8b 4c 24 30
                                                    0x30(%rsp), %ecx
                                            mov
                 83 f9 0a
  40057f:
                                                    $0xa, %ecx
                                            cmp
                 Of 8f a9 00 00 00
                                                    400631 <dud1+0xf1>
  400582:
                                            <del>jg</del>
                 31 f6
                                                    %esi,%esi
  400588:
                                            xor
                 66 Of 1f 44 00 00
  40058a:
                                                    0x0(%rax, %rax, 1)
                                            nopw
                 41 8b 54 b4 04
  400590:
                                                    0x4(%r12,%rsi,4),%edx
                                            mov
  400595:
                 83 fa 08
                                            cmp
                                                    $0x8, %edx
  400598:
                 77 26
                                                    4005c0 <dud1+0x80>
                                            ja
                 89 d0
                                                    %edx, %eax
  40059a:
                                            mov
                 ff 24 c5 50 0b 40 00
  40059c:
                                                    *0x400b50(,%rax,8)
                                            jmpq
  4005a3:
                 83 7c 24 20 01
                                            cmpl
                                                    $0x1,0x20(%rsp)
                                                    400631 <dud1+0xf1>
  4005a8:
                 Of 84 83 00 00 00
                                            <del>je</del>
  4005ae:
                 83 e9 11
                                                    $0x11, %ecx
                                            sub
                 c7 44 24 20 01 00 00
  4005b1:
                                            movl
                                                    $0x1,0x20(%rsp)
  4005b8:
                 00
  4005b9:
                 Of 1f 80 00 00 00 00
                                                    0x0(%rax)
                                            nopl
                                                    $0x1, %ecx
  4005c0:
                 83 e9 01
                                            sub
                                                    $0x1,%rsi
                 48 83 c6 01
  4005c3:
                                            add
  4005c7:
                 48 83 fe 05
                                            cmp
                                                    $0x5,%rsi
                 75 c3
  4005cb:
                                            jne
                                                    400590 <dud1+0x50>
  4005cd:
                 31 c0
                                            xor
                                                    %eax, %eax
  4005cf:
                 83 f9 2a
                                                    $0x2a, %ecx
                                            cmp
  4005d2:
                 Of 94 c0
                                            sete
                                                    %al
  4005d5:
                 48 83 c4 50
                                                    $0x50,%rsp
                                            add
  4005d9:
                 5b
                                                    %rbx
                                            pop
  4005da:
                 5d
                                                    %rbp
                                            pop
  4005db:
                 41 5c
                                                    %r12
                                            pop
  4005dd:
                 с3
                                            retq
  4005de:
                 83 e9 05
                                            sub
                                                    $0x5, %ecx
  4005e1:
                 48 63 c2
                                            movslq %edx, %rax
  4005e4:
                 83 3c 84 01
                                                    $0x1, (%rsp, %rax, 4)
                                            cmpl
  4005e8:
                 74 47
                                                    400631 <dud1+0xf1>
                                            jе
  4005ea:
                 c1 e1 02
                                                    $0x2,%ecx
                                            shl
                 c7 04 84 01 00 00 00
  4005ed:
                                                    $0x1, (%rsp, %rax, 4)
                                            movl
  4005f4:
                 eb cd
                                                    4005c3 <dud1+0x83>
                                            jmp
                 83 7c 24 0c 01
                                            cmpl
  4005f6:
                                                    $0x1,0xc(%rsp)
```

4005fb:	74 34	je	400631 <dud1+0xf1></dud1+0xf1>
4005fd:	01 c9	add	%ecx, %ecx
4005ff:	c7 44 24 0c 01 00 00	movl	\$0x1,0xc(%rsp)
400606:	00		
400607:	eb ba	jmp	4005c3 <dud1+0x83></dud1+0x83>
400609:	83 7c 24 18 01	cmpl	\$0x1,0x18(%rsp)
40060e:	74 21	je	400631 <dud1+0xf1></dud1+0xf1>
400610:	83 c1 07	add	\$0x7,%ecx
400613:	c7 44 24 18 01 00 00	movl	\$0x1,0x18(%rsp)
40061a:	00		
40061b:	eb a6	jmp	4005c3 <dud1+0x83></dud1+0x83>
40061d:	83 7c 24 1c 01	cmpl	\$0x1,0x1c(%rsp)
400622:	74 Od	je	400631 <dud1+0xf1></dud1+0xf1>
400624:	83 c1 03	add	\$0x3,%ecx
400627:	c7 44 24 1c 01 00 00	movl	\$0x1,0x1c(%rsp)
40062e:	00		
40062f:	eb 92	jmp	4005c3 <dud1+0x83></dud1+0x83>
400631:	48 83 c4 50	add	\$0x50,%rsp
400635:	31 c0	xor	%eax,%eax
400637:	5b	pop	%rbx
400638:	5d	pop	%rbp
400639:	41 5c	pop	%r12
40063b:	с3	retq	
40063c:	Of 1f 40 00	nopl	0x0(%rax)

And here is some gdb interaction which should prove useful:

(gdb) x/112x 0x400b40						
0x400b40 <>:	0x0000000	0x0000000	0x25006425	0x00000a64		
0x400b50 <+16>:	0x004005c0	0x0000000	0x004005c0	0x0000000		
0x400b60 <+32>:	0x004005de	0x0000000	0x004005f6	0x0000000		
0x400b70 < +48>:	0x004005e1	0x0000000	0x004005c0	0x0000000		
0x400b80 <+64>:	0x00400609	0x0000000	0x0040061d	0x0000000		
0x400b90 <+80>:	0x004005a3	0x0000000	0x004006c0	0x0000000		
0x400ba0 <+96>:	0x004006c0	0x0000000	0x004006f6	0x0000000		
0x400bb0 <+112>:	0x004006a3	0x0000000	0x0040070a	0x0000000		
0x400bc0 <+128>:	0x004006c0	0x0000000	0x004006de	0x0000000		
0x400bd0 < +144>:	0x004006e1	0x0000000	0x0040071e	0x0000000		
0x400be0 <+160>:	0x004007c0	0x0000000	0x004007c0	0x0000000		
0x400bf0 <+176>:	0x004007f6	0x0000000	0x004007de	0x0000000		
0x400c00 <+192>:	0x004007a3	0x0000000	0x004007c0	0x0000000		
0x400c10 <+208>:	0x004007e1	0x0000000	0x0040080a	0x0000000		
0x400c20 <+224>:	0x0040081d	0x0000000	0x004008c0	0x0000000		
0x400c30 <+240>:	0x004008c0	0x0000000	0x004008a3	0x0000000		
0x400c40 <+256>:	0x004008e1	0x0000000	0x004008de	0x0000000		
0x400c50 <+272>:	0x004008c0	0x0000000	0x004008f6	0x0000000		
0x400c60 <+288>:	0x00400909	0x0000000	0x0040091d	0x0000000		
0x400c70 <+304>:	0x004009b1	0x0000000	0x004009b1	0x0000000		
0x400c80 <+320>:	0x004009cf	0x0000000	0x004009d9	0x0000000		
0x400c90 <+336>:	0x004009d2	0x0000000	0x004009b1	0x0000000		
0x400ca0 <+352>:	0x004009e5	0x0000000	0x004009de	0x0000000		
0x400cb0 <+368>:	0x004009af	0x0000000	0x004009ff	0x0000000		
0x400cc0 <+384>:	0x004009ff	0x0000000	0x00400a1d	0x0000000		
0x400cd0 < +400>:	0x00400a40	0x0000000	0x00400a36	0x0000000		
0x400ce0 <+416>:	0x004009ff	0x0000000	0x00400a30	0x0000000		
0x400cf0 <+432>:	0x00400a29	0x0000000	0x00400a24	0x0000000		
		_				

Hint – don't forget gdb's trick about reversing byte ordering within each 4-byte chunk (same as problem #4).

0 0 000 NUL (null) 1 1 001 SOH (start of heading) 2 2 002 STX (start of text) 3 2 20 040 Space 3 2 1 041 ! ! 6 4 40 100 @ 0 97 61 141 a 2 2 002 STX (start of text) 3 4 22 042 " " 6 6 42 102 B B 98 62 142 b 3 3 003 ETX (end of text) 4 4 004 EOT (end of transmission) 5 5 005 ENQ (enquiry) 6 6 006 ACK (acknowledge) 7 7 007 BEL (bell) 8 8 010 BS (backspace) 9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 32 20 040 Space 44 0100 @ 0 97 61 141 a 65 41 101 A A 64 2102 B B 98 62 142 c 66 42 102 B B 98 62 142 c 66 42 102 B B 98 62 142 c 66 44 104 D D 100 @ O 99 63 143 c 66 44 104 D D 100 @ O 99 63 143 c 67 43 103 C C 99 63 143 c 68 44 104 D D 100 64 144 d 70 46 106 F F 102 66 146 f 70 46 106 F F 102 66 146 f 71 47 107 G G 103 67 147 g 73 49 111 I I 105 69 151 i 74 4A 112 J J 106 6A 152 j 75 4B 113 K K 107 6B 153 k 76 4C 114 L L 108 6C 154 l 77 4D 115 M M 109 6D 155 k 80 50 120 P P 112 70 160 p 81 51 121 Q O 112 64 Q 81 51 121 Q O 112 70 160 p 81 51 124 T T 116 74 164 t 81 51 125 X U 117 75 165 u 81 66 64 104 D D 100 V D 97 61 141 a 84 4 04 4 404 4 48 40; 84 507 4 4 404 4 48 40; 84 507 4 4 404 4 48 40; 84 507 4 4 40 44 4 48 40; 84 508 50 120 V P 102 66 146 l 84 50 10 8 44 50; 84 50 10 8 47 7; 84 61 10 8 47 7; 85 61 14 04 48 40; 86 50 14 04 4 48 30; 99 61 144 4 48 40; 99 61 144 4 48 40; 99 61 144 4 48 40; 99 61 144 4 48 40; 99 61 144 68 150 44; 99 61 140	r:
1 1 001 SOH (start of heading) 2 2 002 STX (start of text) 3 3 21 041 4#33; ! 6 42 102 4#65; A 9 6 6 141 6#78; N 9 6 6 142 102 4#66; B 9 8 62 142 4#99; 3 3 003 ETX (end of text) 3 6 24 044 4#36; 6 6 8 44 104 4#68; D 1 100 64 144 4#100; 5 5 005 ENQ (enquiry) 6 6 006 ACK (acknowledge) 7 7 007 BEL (bell) 8 010 BS (backspace) 9 011 TAB (horizontal tab) 1 B 013 VT (vertical tab) 1 B 013 VT (vertical tab) 1 C 014 FF (NP form feed, new page) 1 3 015 CR (carriage return) 1 E 016 SO (shift out) 1 F 017 SI (shift in) 1 E 010 CQ DLE (data link escape) 1 1 10 020 DLE (data link escape) 1 1 10 020 DLE (device control 1) 1 18 12 022 DC2 (device control 2) 1 13 023 DC3 (device control 2) 1 15 025 NAK (negative acknowledge) 2 10 06 C A#0; 2 17 027 ETB (end of text) 3 2 042 4#34; " 3 42 042 4#34; " 3 42 042 4#36; 6 3 44 104 4#65; A 6 42 102 4#66; B 9 8 62 142 4#99; 66 42 102 4#67; C 67 43 103 4#67; C 68 44 104 4#68; D 100 64 144 4#100; 67 44 107 4#70; F 100 66 145 4#100; 69 45 105 4#69; E 101 65 145 4#101; 69 45 105 4#69; E 101 65 145 4#101; 70 46 106 4#72; H 104 68 150 4#102; 71 47 107 4#71; G 72 48 110 4#73; I 105 69 151 4#105; 73 49 111 4#73; I 105 69 151 4#105; 74 4A 112 4#74; J 106 6A 152 4#106; 75 4B 113 4#75; K 107 6B 153 4#107; 76 4C 114 4#76; L 108 6C 154 4#108; 77 4D 115 4#77; M 109 6D 155 4#107; 78 4E 116 4#76; L 108 6C 154 4#102; 79 4F 117 4#79; O 111 6F 157 4#111; 16 10 020 DLE (data link escape) 17 10 021 DC1 (device control 2) 18 12 022 DC2 (device control 2) 18 12 022 DC2 (device control 2) 18 13 023 DC3 (device control 3) 18 12 022 DC2 (device control 4) 19 13 023 DC3 (device control 4) 19 13 024 DC4 (device control 4) 19 15 025 NAK (negative acknowledge) 19 17 027 ETB (end of trans. block) 10 14 04 4 4#65; B 10 04 4 100 4#68; D 10 04 04 06 6#70; F 10 04 04 04 6#68; D 10 04 04 06 6#70; F 10 04 04 04 6#68; D 100 04 04 04 04 04 04 04 04 04 04 04 04 0	3
2 2 002 STX (start of text) 3 3 003 ETX (end of text) 4 4 004 EOT (end of transmission) 5 5 005 ENQ (enquiry) 6 6 006 ACK (acknowledge) 7 7 007 BEL (bell) 8 8 010 BS (backspace) 9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 22 16 026 SYN (synchronous idle) 23 17 027 ETB (end of trans. block) 34 22 042 6#34; " 35 23 043 6#35; # 36 24 044 6#36; D 36 24 044 6#36; D 37 25 045 6#35; # 38 26 046 6#38; 6 38 26 046 6#38; 6 38 26 046 6#38; 6 38 26 046 6#38; 6 48 104 6#6; D 49 45 105 6#69; E 100 64 144 6#100; 69 45 105 6#69; E 100 65 145 6#101; 70 46 106 6#70; F 102 66 146 6#102; 71 47 107 6#71; G 103 67 147 6#103; 72 48 110 6#72; H 104 68 150 6#104; 73 49 111 6#73; I 105 69 151 6#103; 74 4A 112 6#73; I 105 69 151 6#103; 75 4B 113 6#73; K 107 6B 153 6#109; 75 4B 113 6#77; M 109 6D 155 6#109; 76 4C 114 6#76; L 108 6C 154 6#106; 77 4D 115 6#77; M 109 6D 155 6#109; 110 6F 157 6#111; 110 70 16F 157 6#11; 110 70 16F 157 6#117; 110 70 16F 157 6#117; 110 70 16F 157 6#117; 110 70 16F 6#118; 110 70 16F 6#118; 1110 70 16F 6#118; 111	a
4 4 004 EOT (end of transmission) 5 5 005 ENQ (enquiry) 6 6 006 ACK (acknowledge) 7 7 007 BEL (bell) 8 8 010 BS (backspace) 9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 S0 (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 36 24 044 «#36; \$ 48 44 104 «#68; D 69 45 105 «#69; E 70 46 106 «#70; F 101 69 45 105 «#69; E 101 65 145 «#101; 69 45 105 «#69; E 101 65 145 «#101; 69 45 105 «#69; E 101 65 145 «#101; 69 45 105 «#69; E 102 66 146 «#102; 70 47 107 «#71; G 70 47 107 «#71; G 71 47 107 «#71; G 71 47 107 «#71; G 72 48 110 «#72; H 73 49 111 «#73; I 74 4A 112 «#74; J 75 4B 113 «#75; K 76 4C 114 «#76; L 77 4D 115 «#77; M 78 4E 116 «#78; N 79 4F 117 «#79; O 79 4F 117 «#79; O 79 4F 117 «#79; O 79 4F 117 «#81; Q 79 4F 117 «#81	b
5 5 005 ENQ (enquiry) 6 6 006 ACK (acknowledge) 7 7 007 BEL (bell) 8 8 010 BS (backspace) 9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 37 25 045 % % 48 48 437; % 49 48 106 F F 102 66 146 f 70 46 106 F F 102 66 146 f 70 46 106 F F 102 66 146 f 71 47 107 G G 103 67 147 g 71 47 107 G G 103 67 147 g 72 48 110 H H 104 68 150 h 73 49 111 I I 105 69 151 i 74 4A 112 J J 106 6A 152 j 75 4B 113 K K 107 6B 153 k 76 4C 114 L L 108 6C 154 l 77 4D 115 M M 109 6D 155 m 78 4E 116 N N 110 6E 156 n 79 4F 117 O O 111 6F 157 o 80 50 120 P P 112 70 160 p 81 51 121 Q O 113 71 161 q 81 12 022 DC2 (device control 2) 81 51 121 Q O 113 71 161 q 81 12 022 DC2 (device control 4) 81 51 51 52 U U 117 75 165 u 82 16 026 SYN (synchronous idle) 83 7 067 7 7 87 57 127 W W 119 77 167 w	C
6 6 006 ACK (acknowledge) 7 7 007 BEL (bell) 8 8 010 BS (backspace) 9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 4) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 38 26 046 4#38; 4 39 27 047 4#39; 1 39 27 047 4#31; 1 39 24 044 2C, 054 4#4; 7 34 9 111 4#73; I 104 68 150 4#105; 105 69 151 4#105; 106 6A 152 4#106; 107 6B 153 4#107; 108 6C 154 4#105; 109 6D 155 4#107; 109 6D 155 4#106; 110 6E 156 4#110; 110 6E 156 4#111; 111 021 DC1 (device control 2) 112 70 160 4#112; 113 101 4#4; 113 101 4#4; 114 114 14 14 14 14 14 14 14 14 14 14 14	d
7 7 007 BEL (bell) 39 27 047 6#39; 71 47 107 6#71; G 103 67 147 6#103; 8 8 010 BS (backspace) 40 28 050 6#40; (72 48 110 6#72; H 104 68 150 6#104; 9 9 011 TAB (horizontal tab) 41 29 051 6#41; 73 49 111 6#73; I 105 69 151 6#105; 10 A 012 LF (NL line feed, new line) 42 2A 052 6#42; 74 4A 112 6#74; J 106 6A 152 6#106; 11 B 013 VT (vertical tab) 42 2B 053 6#43; 4 75 4B 113 6#75; K 107 6B 153 6#107; 12 C 014 FF (NP form feed, new page) 44 2C 054 6#44; 76 4C 114 6#76; L 108 6C 154 6#108; 13 D 015 CR (carriage return) 45 2D 055 6#45; - 77 4D 115 6#77; M 109 6D 155 6#109; 14 E 016 SO (shift out) 47 2F 057 6#47; / 78 4E 116 6#78; N 110 6E 156 6#110; 15 F 017 SI (shift in) 47 2F 057 6#47; / 79 4F 117 6#79; O 111 6F 157 6#111; 16 10 020 DLE (data link escape) 48 30 060 6#48; O 17 11 021 DC1 (device control 1) 49 31 061 6#49; 1 81 51 121 6#81; O 112 70 160 6#112; 17 10 12 DC1 (device control 2) 50 32 062 6#50; 2 82 52 122 6#82; R 114 72 162 6#114; 17 15 025 NAK (negative acknowledge) 53 35 065 6#53; 5 85 51 125 6#85; U 117 75 165 6#117; 18 70 160 6#118; 23 17 027 ETB (end of trans. block) 55 37 067 6#55; 7 87 57 127 6#87; W 119 77 167 6#118;	e
8 8 010 BS (backspace) 9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 40 28 050 6#40; (41 29 051 6#41;) 42 2A 052 6#42; * 43 2B 053 6#43; + 44 2C 054 6#43; + 45 2D 055 6#44; , 46 2E 056 6#46; . 47 2F 057 6#47; / 48 110 6#72; H 104 68 150 6#104; 73 49 111 6#73; I 105 69 151 6#105; 107 6B 153 6#107; 74 4A 112 6#74; J 108 6C 154 6#108; 77 4D 115 6#77; M 109 6D 155 6#109; 109 6D 155 6#109; 110 6E 156 6#110; 110 6E 156 6#110; 111 6F 157 6#111; 121 67 166 6#112; 122 16 026 SYN (synchronous idle) 13 03 063 6#51; 3 15 3 063 6#51; 3 16 69 151 6#105; 107 6B 153 6#107; 108 6C 154 6#108; 109 6D 155 6#109; 1	
9 9 011 TAB (horizontal tab) 10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 S0 (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 22 16 026 SYN (synchronous idle) 23 17 027 ETB (end of trans. block) 41 29 051 6#41;) 41 29 051 6#41;) 42 2A 052 6#42; * 42 2A 052 6#42; * 43 2B 053 6#43; + 44 2C 054 6#43; + 45 2D 055 6#45; - 46 2E 056 6#46; . 47 2F 057 6#47; / 48 30 060 6#48; 0 48 30 060 6#48; 0 49 31 061 6#49; 1 50 32 062 6#50; 2 80 55 125 6#82; R 81 51 121 6#81; Q 81 51 121 6#81; Q 82 52 122 6#82; R 83 53 123 6#83; S 84 54 124 6#84; T 85 55 125 6#85; U 86 56 126 6#86; V 87 57 127 6#87; W 87 57 127 6#87; W 87 119 77 167 6#119;	g
10 A 012 LF (NL line feed, new line) 11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 22 16 026 SYN (synchronous idle) 23 17 027 ETB (end of trans. block) 24 2 A 052 * * 42 2A 052 * * 43 2B 053 + + 43 2B 053 + + 44 2 C 054 , , 45 2D 055 - - 46 2E 056 . . 47 2F 057 / / 48 113 K K 107 6B 153 k 47 4D 115 M M 109 6D 155 m 48 110 6E 156 n 49 31 061 1 1 80 50 120 P P 112 70 160 p 81 51 121 Q Q 81 53 123 S S 83 53 123 S S 84 54 124 T T 84 54 124 T T 85 55 125 U U 86 56 126 V V 87 57 127 W W 88 113 K K 87 57 127 W W 88 113 K K 89 107 6B 153 k 86 C 154 l 87 60 155 m 88 60 154 l 89 60 155 m 89 60 155 m 80 60 155 n 80 60 154 l 80 60 155 n 80 60 155 n 80 60 154 l	h
11 B 013 VT (vertical tab) 12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 43 2B 053 + + 44 2C 054 , , 45 2D 055 - - 46 2E 056 . . 47 2F 057 - - 48 113 K K 107 6B 153 k 76 4C 114 L L 108 6C 154 l 77 4D 115 M M 109 6D 155 m 78 4E 116 N N 110 6E 156 n 79 4F 117 O O 111 6F 157 o 80 50 120 P P 112 70 160 p 81 51 121 Q O 113 71 161 q 82 52 122 R R 114 72 162 r 83 53 123 S S 115 73 163 s 84 54 124 T T 116 74 164 t 85 55 125 U U 117 75 165 u 86 56 126 V V 118 76 166 v 87 57 127 W W 119 77 167 w	i
12 C 014 FF (NP form feed, new page) 13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 22 16 026 SYN (synchronous idle) 23 17 027 ETB (end of trans. block) 44 2C 054 , , 45 2D 055 ƽ - 46 2E 056 , , 47 2F 057 / / 48 30 060 0 0 49 31 061 1 1 50 32 062 2 2 31 064 4 4 32 064 4 4 33 065 5 5 34 066 6 6 35 066 6 6 36 56 126 V V 37 119 77 167 w	j
13 D 015 CR (carriage return) 14 E 016 SO (shift out) 15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 23 17 027 ETB (end of trans. block) 45 2D 055 - - 46 2E 056 . . 47 2F 057 / / 48 30 060 0 0 48 30 060 0 0 49 31 061 1 1 50 32 062 2 2 51 33 063 3 3 52 34 064 4 4 53 35 065 5 5 54 36 066 6 6 55 37 067 7 7 57 4D 115 M M 109 6D 155 m 100 6E 156 m 110 6E 156 n 110 6E 156 n 111 6F 157 o 120 020 P P 112 70 160 p 121 020 P P 112 70 160 p 123 023 DC3 (device control 3) 124 024 DC4 (device control 4) 125 025 NAK (negative acknowledge) 126 026 SYN (synchronous idle) 127 027 ETB (end of trans. block) 128 020 055 - - 77 4D 115 M M 109 6D 155 m 120 046 07; N 120 06 08; N 12	k
14 E 016 SO (shift out) 46 2E 056 . . 78 4E 116 N N 110 6E 156 n 15 F 017 SI (shift in) 47 2F 057 / / 79 4F 117 O 0 111 6F 157 o 16 10 020 DLE (data link escape) 48 30 060 0 0 80 50 120 P P 112 70 160 p 17 11 021 DC1 (device control 1) 49 31 061 1 1 81 51 121 Q Q 113 71 161 q 18 12 022 DC2 (device control 2) 50 32 062 2 2 82 52 122 R R 114 72 162 r 19 13 023 DC3 (device control 3) 51 33 063 3 3 83 53 123 S S 115 73 163 s 20 14 024 DC4 (device control 4) 52 34 064 4 4 84 54 124 T T 116 74 164 t 21 15 025 NAK (negative acknowledge) 53 35 065 5 5 85 55 125 U U 117 75 165 u 22 16 026 SYN (synchronous idle) 54 36 066 6 6 86 56 126 V V 118 76 166 v 23 17 027 ETB (end of trans. block) 55 37 067 7 7 87 57 127 W W 119 77 167 w	
15 F 017 SI (shift in) 16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 22 16 026 SYN (synchronous idle) 23 17 027 ETB (end of trans. block) 47 2F 057 6#47; / 48 30 060 6#48; 0 48 30 060 6#48; 0 49 31 061 6#49; 1 50 32 062 6#50; 2 50 32 062 6#50; 2 50 32 062 6#50; 2 51 33 063 6#51; 3 52 34 064 6#52; 4 53 35 065 6#53; 5 54 36 066 6#54; 6 55 37 067 6#55; 7 56 57 127 6#87; W 119 77 167 6#119;	m
16 10 020 DLE (data link escape) 17 11 021 DC1 (device control 1) 18 12 022 DC2 (device control 2) 19 13 023 DC3 (device control 3) 20 14 024 DC4 (device control 4) 21 15 025 NAK (negative acknowledge) 22 16 026 SYN (synchronous idle) 23 17 027 ETB (end of trans. block) 48 30 060 0 0 49 31 061 1 1 50 32 062 2 2 50 32 062 2 2 51 33 063 3 3 52 34 064 4 4 53 35 065 5 5 54 36 066 6 6 55 37 067 7 7 68 56 126 V V 68 57 127 W W 70 160 p 81 51 121 Q Q 113 71 161 q 82 52 122 R R 114 72 162 r 83 53 123 S S 115 73 163 s 84 54 124 T T 116 74 164 t 85 55 125 U U 117 75 165 u 86 56 126 V V 118 76 166 v 87 57 127 W W 119 77 167 w	n
17 11 021 DC1 (device control 1)	0
18 12 022 DC2 (device control 2) 50 32 062 2 2 82 52 122 R R 114 72 162 r 19 13 023 DC3 (device control 3) 51 33 063 3 3 83 53 123 S \$3 115 73 163 s 20 14 024 DC4 (device control 4) 52 34 064 4 4 84 54 124 T T 116 74 164 t 21 15 025 NAK (negative acknowledge) 53 35 065 5 5 85 55 125 U U 117 75 165 u 21 16 026 SYN (synchronous idle) 54 36 066 6 6 86 56 126 V V 118 76 166 v 23 17 027 ETB (end of trans. block) 55 37 067 7 7 87 57 127 W W 119 77 167 w	
19 13 023 DC3 (device control 3)	d
20 14 024 DC4 (device control 4) 52 34 064 4 4 84 54 124 T T 116 74 164 t 21 15 025 NAK (negative acknowledge) 53 35 065 5 5 85 55 125 U U 117 75 165 u 22 16 026 SYN (synchronous idle) 54 36 066 6 6 86 56 126 V V 118 76 166 v 23 17 027 ETB (end of trans. block) 55 37 067 7 7 87 57 127 W W 119 77 167 w	
21 15 025 NAK (negative acknowledge) 53 35 065 5 5 85 55 125 U U 117 75 165 u 22 16 026 SYN (synchronous idle) 54 36 066 6 6 86 56 126 V V 118 76 166 v 23 17 027 ETB (end of trans. block) 55 37 067 7 7 87 57 127 W W 119 77 167 w	3
22 16 026 SYN (synchronous idle) 54 36 066 6 6 86 56 126 V V 118 76 166 v 23 17 027 ETB (end of trans. block) 55 37 067 7 7 87 57 127 W W 119 77 167 w	t
23 17 027 ETB (end of trans. block) 55 37 067 6#55; 7 87 57 127 6#87; ₩ 119 77 167 6#119;	
	W
24 18 030 CAN (cancel) 56 38 070 8 8 88 58 130 X X 120 78 170 x	x
25 19 031 EM (end of medium) 57 39 071 6#57; 9 89 59 131 6#89; Y 121 79 171 6#121;	
26 1A 032 SUB (substitute) 58 3A 072 6#58; 90 5A 132 6#90; Z 122 7A 172 6#122;	Z
27 1B 033 ESC (escape) 59 3B 073 ; ; 91 5B 133 [[123 7B 173 {	{
28 1C 034 FS (file separator) 60 3C 074 < < 92 5C 134 \ \ 124 7C 174	1
29 1D 035 GS (group separator) 61 3D 075 = = 93 5D 135]] 125 7D 175 }	
30 1E 036 RS (record separator) 62 3E 076 > > 94 5E 136 ^ ^ 126 7E 176 ~	
31 1F 037 US (unit separator) 63 3F 077 ? 95 5F 137 _ 127 7F 177	DEL

Answer Sheet

			Name:		
1.	Y	N	5. Fill in all blanks below		
	Y	N	Address (in hex) of the assembly instruction		
	Y	N	that compares i <size calls="" in="" loop="" scanf:<="" td="" that="" the=""></size>		
	Y	N			
2.			Address (in hex) of the assembly instruction that compares a[0]>MAX_FIRST:		
3.	a.				
	b.				
	c.		Address (in hex) of the assembly instruction that does the comparison to produce the value returned by the return statement at the end of the C code (i.e. return()):		
4.	Fill	in all blanks below			
Is the struct with the requested value a leaf node in the tree (i.e. no children)? (Y/N)			Starting address for the instructions that make up the <i>default</i> case of the switch statement (absolute address in hex):		
		es (in hex) of struct with			
the r	equeste	ed value:	Input numbers to return a 1:		
		abel field of the struct with ed value):			