

- HONOR CODE
- Questions Sheet.
- A Lets C. [6 Points]
 - 1. What type of address (heap,stack,static,code) does each value evaluate to Book1, Book1->name, Book1->author, &Book2? [4]
 - 2. Will all of the print statements execute as expected? If NO, write print statement which will not execute as expected?[2]
- B. Mystery [8 Points]
 - 3. When the above code executes, which line is modified? How many times? [2]
 - 4. What is the value of register a6 at the end ? [2]
 - 5. What is the value of register a4 at the end ? [2]
 - 6. In one sentence what is this program calculating ? [2]
- C. C-to-RISC V Tree Search; Fill in the blanks below [12 points]
- D. RISCV - The MOD operation [8 points]
 - 19. The data segment starts at address 0x10000000. What are the memory locations modified by this program and what are their values ?
- E Floating Point [8 points.]
 - 20. What is the smallest nonzero positive value that can be represented? Write your answer as a numerical expression in the answer packet? [2]
 - 21. Consider some positive normalized floating point number where p is represented as: What is the distance (i.e. the difference) between p and the next-largest number after p that can be represented? [2]
 - 22. Now instead let p be a positive denormalized number described as $p = 2^y \times 0.\text{significand}$. What is the distance between p and the next largest number after p that can be represented? [2]
 - 23. Sort the following minifloat numbers. [2]
- F. Numbers. [5]
 - 24. What is the smallest number that this system can represent 6 digits (assume unsigned) ? [1]
 - 25. What is the largest number that this system can represent (assume unsigned). Expression is sufficient ? [1]
 - 26. Convert 122_{10} to unsigned base 4. [1]
 - 27. 4s complement [1]
 - 28. Signed 4s. [1]
- G. Pointer Games [5]
 - 29. Line 5: `x[strlen(x)] = '/ 0'; /* _1 _ */`
 - 30. Line 9: `printf("% d\n", *p); /** 2 **/`
 - 31. `printf("% s\n", cpy); /** 3 **/`

- 32. `printf("%s\n", a());` **4**
- 33. `printf("%s\n", b());` **5**

HONOR CODE

- I have not used any online resources during the exam.
- I have not obtained any help either from anyone in the class or outside when completing this exam.
- No sharing of notes/slides/textbook between students.
- NO SMARTPHONES.

Questions Sheet.

Read all of the following information before starting the exam:

- For each question fill out the appropriate choice or write text on Canvas page. Also type clearly on in the exam on the appropriate text.
- IF THE MULTIPLE CHOICE ANSWER IS WRONG WE WILL MARK THE ANSWER WRONG. IF THE MULTIPLE-CHOICE ANSWER IS CORRECT, WE WILL READ THE WRITTEN PORTION.
- Show all work, clearly and in order, if you want to get full credit.
- I reserve the right to take off points if I cannot see how you logically got to the answer (even if your final answer is correct).
- Circle or otherwise indicate your final answers.
- Please keep your written answers brief; be clear and to the point.
- I will take points off for rambling and for incorrect or irrelevant statements. This test has six problems.

A Lets C. [6 Points]

```

1 // You can assume the appropriate #includes have been done.
2 typedef struct Book{
3     char *name;
4     char *author;
5 } Book;
6
7 Book * createBook() {
8     Book* Book = (Book*) malloc(sizeof(Book));
9     Book->name = "this old dog";
10    char author[100] = "mac demarco";
11    Book->author = author;return Book;
12 }
13 int main(int argc, char **argv) {
14     Book *Book1 = createBook();printf("%s\n", "Book written:");printf("%s\n", Book1->name);
15     // print statement #1
16     printf("%s\n", Book1->author);
17     // print statement #2
18     Book Book2;
19     Book2.name = malloc(sizeof(char)*100);
20     strcpy(Book2.name, Book1->name);
21     Book2.author = "MAC DEMARCO";
22     printf("%s\n", "Book written:");
23     printf("%s\n", Book2.name);
24     // print statement #3
25     printf("%s\n", Book2.author);
26     // print statement #4
27     return 0;
28 }

```

1. What type of address (heap,stack,static,code) does each value evaluate to Book1, Book1->name, Book1->author, &Book2? [4]

Book1: Heap

Book1->title: Static

Book1->author: Stack

&Book2: Stack

2. Will all of the print statements execute as expected? If NO, write print statement which will not execute as expected?[2]

print statement #2

B. Mystery [8 Points]

```

1 | .globl main
2 |
3 | .text
4 | main:
5 | li a0,1
6 | li a1,0
7 |
8 | la s0, L1
9 | lw s1, 8(s0)
10 | addi s2, zero, 6
11 | addi s3, zero, 0
12 | li s4,8449
13 | slli s4, s4,7
14 |
15 | L1:
16 | beq s3, s2, done
17 | add s1, s1, s4
18 | add a2, a0, a1
19 | sw s1, 8(s0)
20 | addi s3, s3, 1
21 | j L1
22 |
23 | done:
24 | li a0,10
25 | ecall

```

3. When the above code executes, which line is modified? How many times? [2]

Line 18: add a2,a0,a1 6 times.

4. What is the value of register a6 at the end ? [2]

8. first iteration line 18 is add a2,a0,a1 . Next iteration it is
add a3,a1,a2 , next iteration add a4,a2,a3 ... and so on.

5. What is the value of register a4 at the end ? [2]

3

6. In one sentence what is this program calculating ? [2]

Fibonacci series

\pagebreak

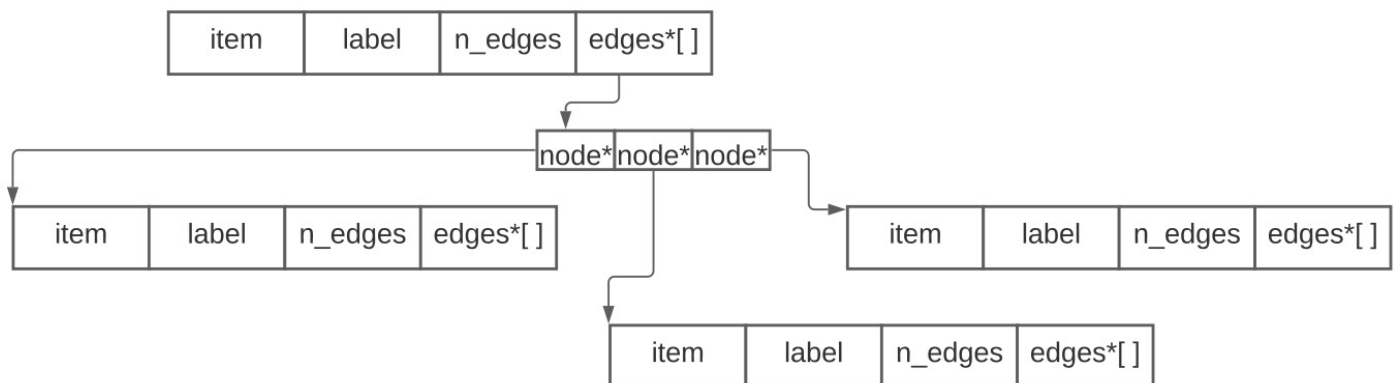
C. C-to-RISC V Tree Search; Fill in the blanks below [12 points]

We're interested in running a search on a tree, and labeling the nodes in the order we finish examining them. Below we have the struct definition of a node in the tree, and the implementation of the function in C.

**Note that initially, all nodes in the graph have their label set to -1. The address width of our machine is 32 bits.

**

```
1 struct node {
2     int item;
3     int label;
4     int n_edges;
5     struct node** edges[];
6 }
7
8 int label(struct node* nd, int counter) {
9
10    if (nd->label != -1) {
11        return counter;
12    }
13
14    for (int i = 0; i < nd->n_edges; ++i) {
15        counter = label(nd->edges[i], counter);
16    }
17
18    nd->label = counter++;
19    return counter;
20 }
```



Fill in the blanks in the code below

```

1  dfs_label:
2  # prologue
3  addi sp sp, -12 # Q7
4  # Q1. We need to make space for 3 registers on stack.
5  # 3*4bytes/register = 12
6  sw ra, 0(sp)
7  sw s0, 4(sp)
8  sw s1, 8(sp)
9  sw s2, 12(sp)
10
11 # a1 is counter. We do not need temporaries for counter
12 # since it is always overwritten
13 # by a call. a0 is node*
14 # Base case
15 addi t1, zero, -1 # Q8: Base case load immediate -1
16 lw t0, 4(a0) # Q9 Load n->label for comparison to base case.
17 add a0,a1,zero # Q10. Why have this instruction?
18 bne t0, t1, epilogue
19
20 # Loop header
21 add s0, a0, zero # Q11 Save node in a saved register.
22 # a0 is going to be eventually overwritten.
23 addi s1, zero, 0
24
25 loop:
26 lw t0 _8(s0)_ # Q12 Load in n_edges in t0
27 beq t0, s1, fin # Q13 Iterator. If s1 end of loop iteration. We stop
28 lw a0, 12(s0) # Q14 load edges into a0
29 sll t0, s1, 2 # Q15 Shift by 2 to get byte offset
30 add a0, a0, t0 # Get ptr for next node
31 lw a0, 0(a0) # Q16 Load ptr into a0
32 jal label
33 addi a1, a0, 0 # Q17 Put return value into second argument for next call.
34 addi s1, s1, 1 # Increment counter after returning from recursion
35 j loop
36
37 fin:
38 sw a1 _4(s0)_ # Q18.
39 addi a1, a1, 1
40 # counter++;
41 # you have to increment after storing the value.
42 addi a0, a1, 0
43
44 epilogue:
45 lw ra (sp)
46 lw s0 4(sp)
47 lw s1 8(sp)
48 add sp, sp, 12
49 jr ra

```

\pagebreak

D. RISCV - The MOD operation [8 points]

We will be introducing a new instruction called "mod" in RISC-V which calculates the remainder. The semantics of the mod instruction (`mod dst,src1,src2`) are $dst = src1 \% src2$. e.g., lets say $s1=5$ $s2=2$
`mod s3,s1,s2` stores the value 2 in s3.

We will be using the mod operation in the program below called cipher.

You want to impress your friend, so you predict the result of executing the program as it is written, just by looking at it. If the program is guaranteed to execute without crashing, describe what it prints, otherwise explain the bug that may cause a crash.

```

.globl main
.data
a: .string "happy"
init: .string "XXXXX" # 12 Xs

.text

# C : Cipher(char* str).

cipher:
addi s3,zero,25

loop_header:
lbu s2, 0(a0) # Read character ch
beqz s2, end
addi s7,a0,0
addi a0,a0,1

loop:
addi s1, s2, -97 #
bltu s3,s1,loop_header
addi s2,s2,13
mod s2,s2,26 # New instruction
addi s2,s2,97
sb s2, 0(a1)
addi a1,a1,1
jal loop_header
end:
ret

main:
la a0, a
la a1,init
jal cipher
li a0,10
ecall

```

19. The data segment starts at address 0x10000000. What are the memory locations modified by this program and what are their values ?

ngvve : 0x10000006 0x1000000A

E Floating Point [8 points.]

The TAs get tired of having to convert floating-point values into 32 bits. As a result they propose the following smaller floating-point representation which is useful in a number of machine learning applications. It consists of a total of 8 bits as show below.

Sign	Exponent	Mantissa
1 bit	3 bits. Bias 3.	4 bits.

Exponent bias is 3.

Exponent	Significand	Meaning
0	0	0
7	non-zero	NaN
7	0	+ - inf
0	non-zero	Denorm

- The largest exp remains reserved as in traditional floating point
- The smallest exp follows the same denormalized formula as traditional floating point
- Numbers are rounded to the closest representable value. Any numbers that have 2 equidistant representations are rounded down towards zero.

20. What is the smallest nonzero positive value that can be represented? Write your answer as a numerical expression in the answer packet? [2]

Exponent - 0

.0001 - Denormalized

$$= 2^{-2} * 0.0001 = 2^{-6}$$

21. Consider some positive normalized floating point number where p is represented as: What is the distance (i.e. the difference) between p and the next-largest number after p that can be represented? [2]

$$p = 2^y \times 1.\text{significand}.$$

$$2^{(y-4)}$$

22. Now instead let p be a positive denormalized number described as $p = 2^y \times 0.\text{significand}$. What is the distance between p and the next largest number after p that can be represented? [2]

$$2^{-6}$$

23. Sort the following minifloat numbers. [2]

0x04, 0xb2, 0x62, 0x45, 0x32

0x32 (1.125), 0xb2 (-1.125), 0x45 (2.625), 0x62 (8.75), 0x04 (0.0625)

0xb2 < 0x04 < 0x32 < 0x45 < 0x62

F. Numbers. [5]

We are going to be creating a new base system for numbers based on 4s.

So every number is going to be represented as a power of 4. A base 4 system has 3 symbols at most (0...3). e.g., 22_4 is equal to $2 \times 4^0 + 2 \times 4^1 = 2 + 8 = 10_{10}$.

24. What is the smallest number that this system can represent 6 digits (assume unsigned) ? [1]

0

25. What is the largest number that this system can represent (assume unsigned). Expression is sufficient ? [1]

$$4092. 4^6 - 4$$

$$333333_4$$

$$\sum_{i=0}^5 3 \times 4^i$$

$$3 \times 4^5 + 3 \times 4^4 + 3 \times 4^3 + 3 \times 4^2 + 3 \times 4^1 + 3 \times 4^0 = 3 \times (1024 + 256 + 64 + 16 + 4 + 1) =$$

26. Convert 122_10 to unsigned base 4. [1]

$$122 = 1 \times 4^3 + 3 \times 4^2 + 2 \times 4^1 + 2 \times 4^0$$

$$1322_4$$

27. 4s complement [1]

Recall 2s complement where we add a bias to represent -ve numbers.

Consider a 4 digit 2s complement system. In unsigned form it can represent all numbers in range 0...15. In 2s complement form we use a negative bias to represent integers in the negative range. For instance in 2s complement with 4 digit numbers we use a negative bias of -16.

Decimal Number	Two's Complement
-8	1000
-7	1001

Decimal Number	Two's Complement
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Suppose we wanted to use a 4s complement notation. If we are working with a 4 digit base 4 number, what should we choose as our bias? Our bias should create roughly equal amounts of negative and positive numbers for our range.

Answer: 4^4

28. Signed 4s. [1]

For each number, we will reserve the most significant digit to strictly be used as a sign bit [1: negative 0:positive]. (e.g., 0001: +1, 1020: -8, 2020 - Not valid sign bit something other than 1/0). What is the number of numbers that can be represented using this notation.

Answer : $2 \cdot 4^3 = 128$

G. Pointer Games [5]

```
1 char *feels;
2 char *message(char *msg) {
3     char *x = malloc(sizeof(char) * (strlen(msg) + 1));
4     strncpy(x, msg, strlen(msg));
5     x[strlen(x)] = '\0'; / ****1 * ** /
6     return x;
7 }
8 void p_int(int *p) {
9     printf("%d\n", *p); /**** 2 ****/
10 }
11
12 void p_msg(char *str) {
13     char *cpy = calloc(strlen(str) + 1, 1);
14     strncpy(cpy, str, strlen(str));
15     printf("%s\n", cpy); /**** 3 ****/
16 }
17 char *a() {
18     char res[7] = "rules";
19     return res;
20 }
21 char *b() {
22     char *var = "cmpt295";
23     return var;
24 }
25 void c() {
26     printf("%s\n", a()); /**** 4 ****/
27     printf("%s\n", b()); /**** 5 ****/
28 }
29 int main() {
30     int y;
31     feels = malloc(3);
32     strcpy(feels, "hi");
33     message(feels);
34     p_int(&y);
35     p_msg(feels);
36     c();
37 }
```

There are comments on lines with numbers from 1-5. Each of these lines dereference a pointer. Characterize if these memory accesses are legal c. We will use the following terminology

- Legal:
- **Initialized:** Is there actual meaningful data in contents (data at each address) or is it garbage.
- **Illegal:** This line will always dereference an address the program doesn't have explicit access to

- **Possibly Legal:** The operation could result in only dereferences of legal addresses but it's also possible that in other runs on the program illegal accesses occur.

Mark which of the following apply to questions below:

29. Line 5: `x[strlen(x)] = '/ 0';` / * _1 _ * /

Possibly legal.

30. Line 9: `printf("% d\n", *p);` / 2 **/**

Legal, but uninitialized.

31. `printf("% s\n", cpy);` / 3 **/**

Legal and initialized

32. `printf("% s\n", a());` / 4 **/**

illegal

33. `printf("% s\n", b());` / 5 **/**

legal and initalized.