- HONOR CODE
- Questions Sheet.
- A. Easy. 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. Medium. 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. Hard. C-to-RISC V Tree Search. Fill in the blanks below [12 points]
- D. Easy. 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. Medium. 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 asp = 2^y x 0.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. Easy. 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. Easy. 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", a()); /** **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.
- **CANVAS ANSWERS WILL BE LOCKED AFTER 1ST TRY.**

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 answeris 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. Easy. Lets C. [6 Points]

```c
// You can assume the appropriate #includes have been done.
typedef struct Book{
  char *name;
  char *author;
} Book;

Book * createBook() {
    Book* Book = (Book*) malloc(sizeof(Book));
    Book->name = "this old dog";
    char author[100] = "mac demarco";
    Book->author = author;
    return Book;
}
int main(int argc, char **argv) {
    Book *Book1 = createBook();
    printf("%s\n", "Book written:");
    printf("%s\n", Book1->name);    // print statement #1
    printf("%s\n", Book1->author);  // print statement #2
    Book Book2;
    Book2.name = malloc(sizeof(char)*100);
    strcpy(Book2.name, Book1->name);
    Book2.author = "MAC DEMARCO";
    printf("%s\n", "Book written:");
    printf("%s\n", Book2.name);     // print statement #3
    printf("%s\n", Book2.author);   // print statement #4
    return 0;
}
```

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. Medium. 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]

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. Hard. 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.
**

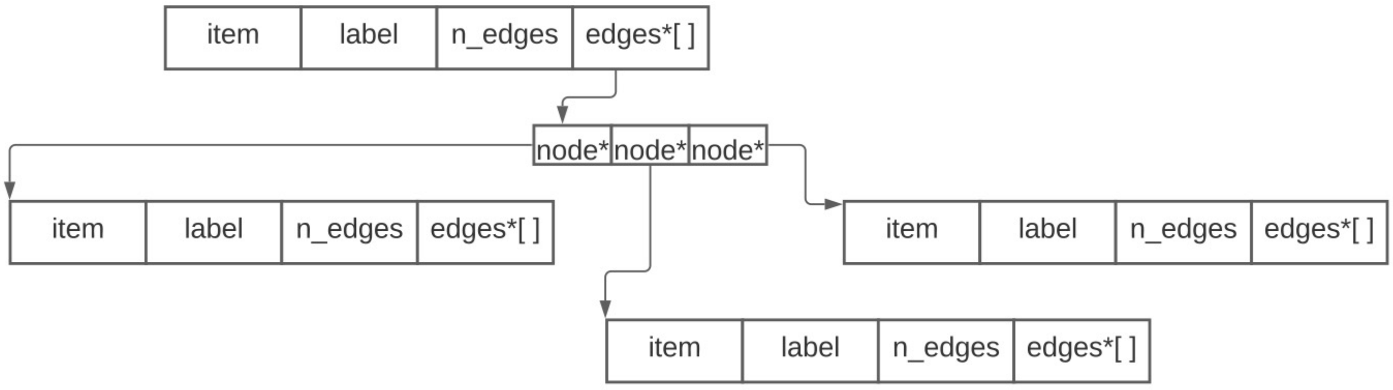
```c
struct node {
  int item;
  int label;
  int n_edges;
  struct node** edges[];
}

int label(struct node* nd, int counter) {

  if (nd->label != -1) {
    return counter;
  }

  for (int i = 0; i < nd->n_edges; ++i) {
    counter = label(nd->edges[i], counter);
  }

  nd->label = counter++;
  return counter;
}
```



Fill in the blanks in the code below

```
label:
# prologue
addi sp sp, -Q7-- Fill in immediate
sw ra, 0(sp)
sw s0, 4(sp)
sw s1, 8(sp)

# a1 is counter. We do not need temporaries for counter
# since it is always overwritten
# by a call. a0 is node*
# Base case
addi t1, zero, --- # Q8: Fill in immediate
------Q9-----------  # Q9 Fill in instruction.
addi a0,a1,zero      # Q10 Why do we have this instruction?
bne t0, t1, epilogue

# Loop header
add s0, ----Q11---- # Fill the instruction; What is s0 used used for?
# a0 is going to be eventually overwritten.
addi s1, zero, 0 # Set loop index to 0

loop:
lw t0 ____Q12_____
beq ____Q13_____
lw a0, ____Q14_____
sll t0, s1, ____Q15_____
addu a0 a0 t0  # Get ptr for next node
lw a0, _____Q16______
jal label
  addi a1, ____Q17_____  # Put return value into second argument for next call.
addi s1, s1, 1 # Increment counter after returning from recursion
j loop

fin:
sw a1 __Q18__
addi a0, a1, 0 # counter++;
# you have to increment after storing the value.
# Return counter
addi a0, a1, 0

epilogue:
lw ra (sp)
lw s0 4(sp)
lw s1 8(sp)
add sp, sp, 12
jr ra
```

D. Easy. 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" # 5 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 ?

E. Medium. 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]

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 asp = 2^y x 0.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]

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

F. Easy. Numbers. [5]

We are going to be creating a new base system for numbers based on 4s.
So every number is going to 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]

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]

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

Decimal Number	Two's Complement
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111

Decimal Number	Two’s Complement
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.

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:postivie]. (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.

G. Easy. Pointer Games [5]

```
char *feels;
char *message(char *msg) {
  char *x = malloc(sizeof(char) * (strlen(msg) + 1));
  strncpy(x, msg, strlen(msg));
  x[strlen(x)] = ‘/ 0’; / ****1 * *** /
  return x;
}
void p_int(int *p) {
  printf("% d\n", *p); /**** 2 ****/
  }

void p_msg(char *str) {
  char *cpy = calloc(strlen(str) + 1, 1);
  strncpy(cpy, str, strlen(str));
  printf("% s\n", cpy); /**** 3 ****/
}
char *a() {
  char res[7] = " rules";
  return res;
}
char *b() {
  char *var = "cmpt295";
  return var;
}
void c() {
  printf("% s\n", a()); /**** 4 ****/
  printf("% s\n", b()); /**** 5 ****/
}
int main() {
  int y;
  feels = malloc(3);
  strcpy(feels, "hi");
  message(feels);
  p_int(&y);
  p_msg(feels);
  c();
}
```

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

- Legal:
- **Initialized**: Is there actual meaningful data
-

- **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 _ * /

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

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

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

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