Name: ______________________ Entry num: ______________________

***[9 marks]* IO virtualization.**
For these problems, assume a simple trap-and-emulate hypervisor with no dynamic binary translation, etc.

[1 mark] (T/F) Writing to a virtualized IO device register causes a VM exit for both Port IO devices and MMIO devices.

____

[1 mark] (T/F) Reading from a virtualized IO device register may cause no VM exits for MMIO devices.

____

[1 mark] (T/F) Reading from a virtualized IO device register may cause no VM exits for Port IO devices.

____

[2 marks] Mark each network IO virtualization technique with 1, 2, and 3.

The technique marked with 1 can be expected to provide the *highest* throughput. The technique marked with 3 is expected to provide the *least* throughput.

____ Full virtualization: Emulating e1000, a real physical NIC.
____ Paravirtualization: virtio-net
____ Direct device assignment with SRIOV NICs

[4 marks] In the direct device assignment, the guest device driver wants to give access to GPA=X, which translates to HPA=Y, to the device. How might a 2-D IOMMU be set up for this?

***[10 marks]* Primary backup replication.**

[2 mark] Mark each workload with 1, 2, and 3.

The workload marked with 1 can be expected to experience the *least* slowdown when run in fault-tolerant virtual machines compared to when they are run in a single VM. The workload marked with 3 is expected to experience the *most* slowdown.

______ Computing Nth fibonacci number.
______ Sending 1GB of data to a client.
______ Receiving 1GB of data from a client.

[3 marks] Justify your answers to the previous question.

[5 marks] Let's say that the shared storage server accepts only two types of requests (other than the test-and-set request used during failover):

- <read, block number>: This returns data in the requested disk block number.
- <append, block data>: This internally maintains a "current block number", writes data to the current block number, and then increments the current block number.

You can further assume that the VM is appending disk blocks one after another sequentially in a simple loop.

Can fault-tolerant VMs create an unintended disk state? If yes, describe the exact sequence of events and their unintended outcome. Further, describe how the APIs and their behaviours can be changed so that the disk state remains correct. Note that you should still support read and append APIs. Their parameters and return values may be modified. New API may be added to support the read and append APIs.

[10 marks] **Distributed computation**

Let us say that we are given the following Spark program to compute the transitive closure of a graph.

```
spark = # Initialize Spark
tc = # Input a graph as a collection, i.e, RDD, of (src, dst) tuples

#Linear transitive closure: each round grows paths by one edge, by
#joining the graph's edges with the already-discovered paths. e.g.
#join the path (y, z) from the TC with the edge (x, y) from the
#graph to obtain the path (x, z).

#Because join() joins on keys, the edges are stored in a reversed
order.
edges = tc.map(lambda x_y: (x_y[1], x_y[0]))

oldCount = 0
nextCount = tc.count()

while True:
      oldCount = nextCount

      #Perform the join, obtaining an RDD of (y, (z, x)) pairs,
      #then project the result to obtain the new (x, z) paths.
      joined = tc.join(edges)
      new_edges = joined.map(lambda x: (x[1][1], x[1][0]))
      tc = tc.union(new_edges).distinct()

      nextCount = tc.count()
      if nextCount == oldCount:
            Break

print("TC has %i edges" % tc.count())
spark.stop()
```

Name: ______________________ Entry num: ______________________

[4 marks] Assuming that the program completes in two iterations across the while loop. Draw the lineage graph of this program.

[3 marks] In the lineage graph drawn above, identify RDDs that you might want to checkpoint. Justify your answer.

[3 marks] Assume that the input graph, `tc,` is partitioned by `src` in the `(src, dst)` tuple. Mark narrow and wide dependencies in the lineage graph.

***[11 marks]* Raft**

Let us say that the workers **a**, **b**, **c**, **d**, and **e** have the following logs. Each entry in a worker's log just specifies the term number.

Log index ->	***0***	***1***	***2***	***3***	***4***	***5***
a's log ->	1	4	4	5		
b's log ->	1	4				
c's log ->	1	4	4	5		
d's log ->	1	2	2	3	3	
e's log ->	1	4	4	4	4	

[2 marks] Circle the log entries in the table above that may be committed entries.

[6 marks] Specify a leader election trace that may lead workers into these logs. In particular, you have to answer which worker may have become leader in which term, who may have voted for that leader, when did the leader crash, etc.

[3 marks] Draw a table indicating which worker may vote for which workers in term 6 election? Also identify which workers may become leader in term 6.

[15 marks] **A simple microservice**

Let us say that we have the following hypothetical microservice.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int exponent(int a, int b) {
  int result = 1;
  for(int i=0; i< b; i++)
    result *= a;
  return result;
}

int* charToInt(char* arr_ch, int len) {
  int* arr_int = (int*) malloc(len * sizeof(int));
  for(int i = 0; i < len; i++) {
    arr_int[i] = (int)(arr_ch[i]);
  }
  return arr_int;
}

void getTime(char* time_str) {
  time_t current_time;
  time(&current_time);

  struct tm* time_info = localtime(&current_time);
  strftime(time_str, sizeof(time_str), "%H:%M:%S", time_info);
}

int main() {
  char time_str[9];
  getTime(time_str);

  printf("%s: %d", time_str, exponent(3, 4));
  return 0;
}
```

When the service is run, it just logs the current time and `exponent(3, 4)` to the serial console and exits. For example, it prints the following when it is run three times, once every second.

```
11:53:50: 81
11:53:51: 81
11:53:52: 81
```

Name: ______________________ Entry num: ______________________

[3 marks] For the program above (assuming no compiler optimizations), draw the unikraft dependency graph. Please specify any assumptions you make about unikraft. Briefly justify each edge in your dependency graph.

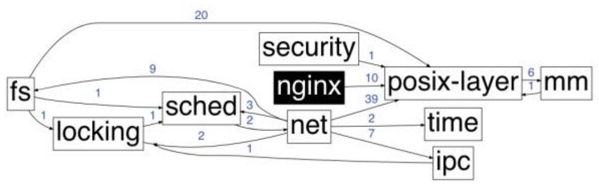

Figure 2. Nginx Unikraft dependency graph

Hints: Following is the Unikraft dependency graph of NGINX, a load balancer. In the figure, `ipc` stands for Inter-process communication, `mm` stands for memory manager, `fs` stands for filesystem, and `net` stands for network.

[3 marks] Rewrite an optimized version of the given C program after applying dead-code elimination and constant folding.

Hints: Dead-code elimination is an optimization that removes code which does not affect the program results. For example, the `if (0) { … }` can just be deleted:

Constant folding is the process of recognizing and evaluating constant expressions at compile time rather than computing them at runtime. For example, the compiler can replace x = 5 + 2 with just x = 7.

[3 marks] For the optimized program above, redraw the Unikraft dependency graph.

[5 marks] Let's say that the VM image was compiled (without the compiler optimizations) for x86 but we would like to run it on a server with ARM CPUs. Describe how one might run it. Please use just the `charToInt` function and describe how it may be run.

[1 mark] Iterative pre-copy live migration will finish in a lesser number of iterations if we are calling the `exponent` function in a loop compared to if we are calling `charToInt` function in a loop.

____

[10 marks] **Funkernels**

Recently, we are observing another shift from microservices to FaaS. Let's say we wish to leverage this and design *Funkernels:* VM images specialised for running just a *single function once*. For the same microservice given above, we are interested in creating three different Funkernels: one for the `exponent` function, one for the `charToInt` function, and one for the `getTime` function.

Further assume that the FaaS hypervisor is able to custom setup registers and memory for each VM image. It can give control to the VM image and finally can do a custom read of registers and memory to know the return value of each function.

Also assume that you have access to the best possible compiler that can apply any code optimization you can think of. Basically, try to make each Funkernel really minimal.

[3 marks] Design a Funkernel for the `exponent` function.

[3 marks] Design a Funkernel for the `charToInt` function.

[4 marks] Suggest an ordering of start times and image sizes between the three VM images: the unikernel for the unoptimized microservice, the funkernel for the `exponent` function, and the funkernel for the `charToInt` function. Justify your answer.

Rough sheet