EECS 678 – Operating Systems – Fall 2020
Quiz – 4

Name: _Prasad Kulkarni_ (Please Print)

1. **True or False (1 point):** The shared memory IPC model cannot be used to communicate between processes on different machines.
   (True is circled)

2. **True or False (1 point):** The OS protects the address space of the parent *thread* from the address space of the child thread.
   (False is circled)

3. **Select the correct answer (1 point):** An exit() called from any thread will kill
   (a) just that thread
   (b) the entire process *(circled)*

4. **Mark all that apply (2 points):** The following process/OS resources are *unique* to each sibling thread:
   (a) heap (dynamically allocated data)
   (b) stack (function activation records) *(circled)*
   (c) open file descriptors
   (d) hardware register state *(circled)*

5. **Mark all that apply (2 points):** The many-to-one multi-threading model has the following properties:
   (a) does not need any (additional) support from the OS *(circled)*
   (b) can run the threads on different cores of a multi-core machine
   (c) one thread blocked (on I/O) blocks all other sibling threads *(circled)*
   (d) terminating one child thread terminates all sibling threads

6. **Answer (1 point):** A 4-byte integer variable is currently holding the value 0x2708a79d. A machine with big endian byte ordering stores this value in 4 consecutive memory bytes as follows: |27|08|a7|9d|
   How will this value be stored in a little-endian machine?

   |9d|a7|08|27|

Please turn page over

7. **Answer (3 points):** The figures below show the code and process address space representation *before the call to pthread_create()*. Illustrate the process address space *after the call to pthread_create()*.
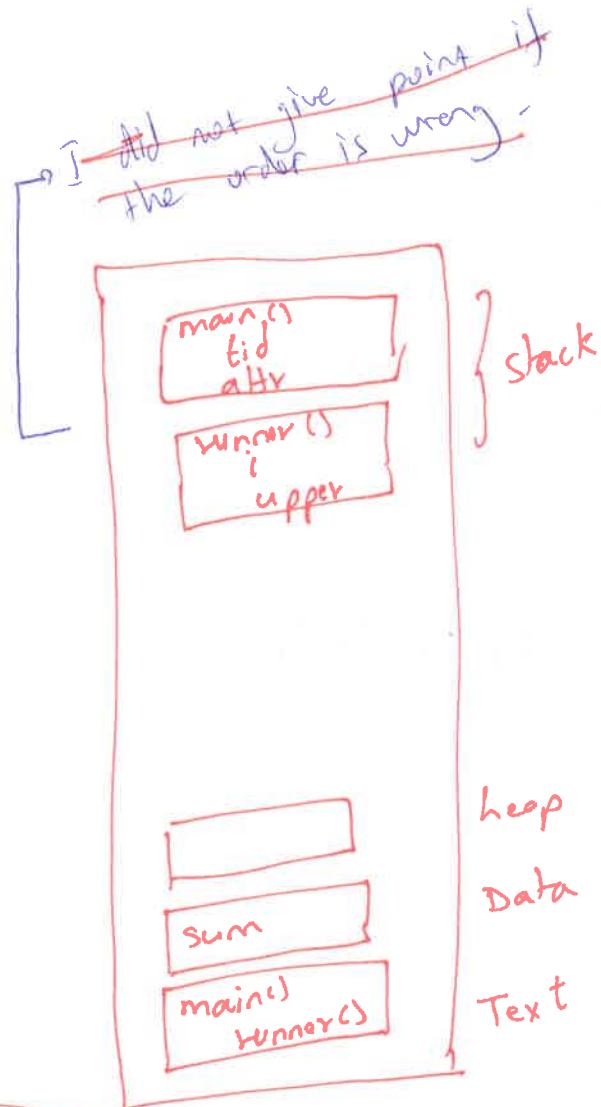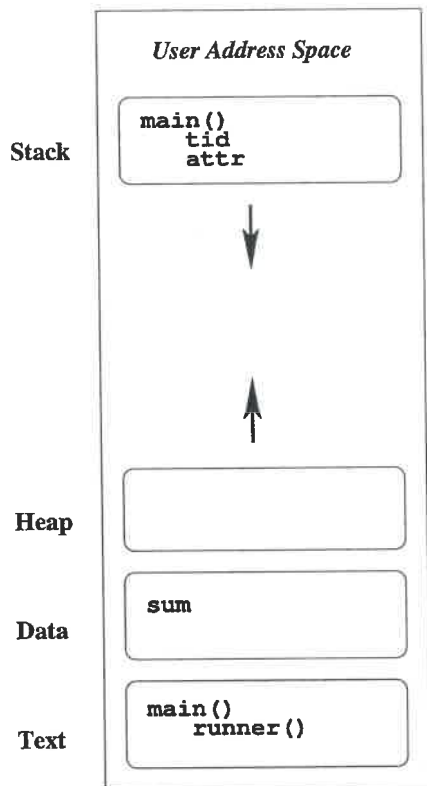
```c
int sum;
int main (int argc, char *argv[])
{
  pthread_t tid; /* thread identifier */
  pthread_attr_t attr /* set of thread attributes */

  pthread_attr_init(&attr);
  pthread_create(&tid, &attr, runner, "100");
  pthread_join(tid, NULL);
}

void *runner (void *param)
{
  int i, upper = atoi(param);

  for(i=1 ; i<=upper ; i++)
      sum += i;
  pthread_exit(0);
}
```

I did not give point if the order is wrong.

User Address Space

Stack — main() / tid / attr

Heap

Data — sum

Text — main() / runner()

[handwritten annotation diagram:]

main() / tid / attr
runner() / i / upper } Stack

Heap

Data — sum

Text — main() / runner()

grading
+1 if heap, data & Text unchanged
+1 if 2 boxes on the stack
+1 variables 'i' and 'upper' placed in 'runner()' activation record.