Problem 1. Revisions

Stack and heap are two ways to help implement the memory allocation, stack is for static memory while heap is usually for dynamic memory.

The stack is the memory set aside for a thread of execution, which is always reserved in a LIFO(last in first out) order. When a function is called, a block is reserved on top of the stack for local variables, when that function returns, that block becomes unused and can be reserved again for the next function calls, this makes the stack much simpler to keep track of. The stack is set to a fixed size and and can not grow past its fixed size, so if there's not enough room on the stack to handle the memory assigned to it, a stack overflow occurs. The stack is much faster than the heap, this is because of the way that memory is allocated on the stack, and the data allocated on the stack is automatically de-allocated when variables go out of the scope.

The heap is the memory set aside for dynamical allocation, which is typically allocated at application startup by the runtime, and is reclaimed when the application quits. Different from the stack is attached to the thread, one can allocate and de-allocate the heap anytime he want. In the language like C or C++, data stored on the heap has to be deleted manually, and it is much more complex to keep track of which part of the heap is allocated or free at a given time.

Problem 2. Personal Research

1. There are several operations performed after the power button of the computer is pressed down. First, the BIOS inside the ROM will go through a Power-On Self-Test, if there is anything wrong with the hardware, the motherboard will buzz and end the booting program. Second, the BIOS will give out its control to the first device on the boot sequence, in this phase, the hard disk is read to load and execute the operating system. After that, the computer is ready to take command by its user.

   The BIOS is used in the first phase check the configuration and hardware condition through its self-test, after that, BIOS will give the full control over the computer to the operating system.

2. - Hybrid kernal is a OS kernal architecture combining micro kernals and monolithic kernals.
   - Exo kernal is a small kernal which gives direct access towards the hardware.

Problem 3. Course Application

1. (a), (c) and (d) should only be allowed in kernel mode.

   (a) Disable all interrupts needs to send request to the kernel.

   (b) Read the time-of-day clock is done in both user mode and kernel mode.

   (c) Set the time-of-day clock belongs to the previleged instructions.

   (d) Change the memory map is an instruction relates to some hardware resources that has to be in done kernel mode.

2. Since there are two CPUs, the most optimal solution is when two programs are executed in parallel, which will cost 20ms in total. Otherwise, 25ms, 30ms, 35ms are all possible answers considering different sequence of the execution of the program.

Problem 4. Simple Problem

1. The video RAM needed to support a 25 lines by 80 rows character monochrome text screen is $25 \times 80/8/1024 = 0.24$KB.
2. The video RAM needed to support a 1024 x 768 pixel 24-bit color bitmap is $1024 \times 768 \times 24/8/1024 = 2304$KB.
3. The first solution cost $0.24 \times 5 = \$1.2$. The second solution cost $2304 \times 5 = \$11520$.
4. Nowadays, the RAM price is about \$5/GB, so the above two solutions will just cost tiny amount of money.

Problem 5. Command lines on a Unix System

Please check *h1.sh* and *README.md*

```
#!/bin/bash
# 1. Create a new user
useradd graves
# 2. List all the currently running processes
ps -e
# 3. Display the characteristics of the CPU and the available memory
top
# 4. Redirect some random output into two different files
shuf -i 0-1000 -n 10 | tee 1.txt >> 2.txt
# 5. Concatenate the two previous files
cat 1.txt 2.txt > 3.txt
# 6. Read the content of the resulting file as hexdecimal values (in
↪  other words find a command to read a file as hexadecimal values)
od -t x1 -An result.txt
# 7. Use a single command to find all the files in /usr/src with the
↪  word semaphore in their name and containing the word ddekit sem
↪  down
find /usr/src -name '*semaphore*' | xargs grep -lw 'ddekit_sem_down'
```