Problem 1. Multiprogramming

1. Since all process spending the same fraction p of their time waiting for Input/Output (I/O) to complete, the probablity for 1 process waiting at some specific time is p, for n processes it will be p^n. Then after waiting, the rest $1 - p^n$ will be the computer utilization.

2. The curve of $1 - p^n$ is sketched below. The blue curve denotes $p = 25\%$, the green curve denotes $p = 60\%$, and the red curve denotes $p = 90\%$

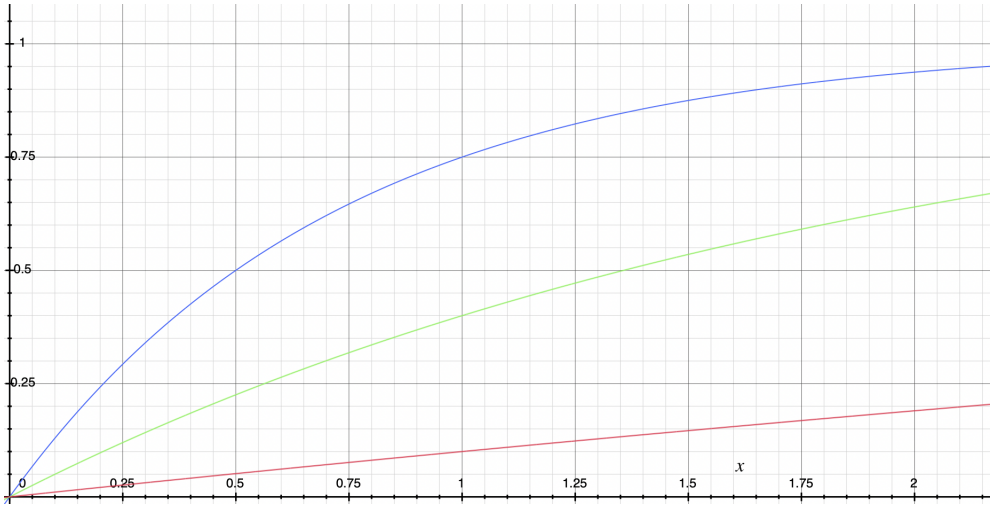

Figure 1: CPU Utilisation

3. (a) Since $(256 - 96)/48 = 3.3$, 3 processes can be stored.

   (b) CPU utilization is calculated as $1 - 0.9^3 = 27.1\%$.

   (c) We need to consider the improvement rate regarding certain amount of RAM added.

   - When 256 MB is added, $\lfloor (512 - 96) \div 48 \rfloor = 8$ processes can be store simultaneously in memory, the CPU utilisation is $1 - 0.9^8 \approx 56.95\%$. It has a improvement of 29.85% per 256 MB.
   - When 512 MB is added, $\lfloor (768 - 96) \div 48 \rfloor = 14$ processes can be store simultaneously in memory, the CPU utilisation is $1 - 0.9^{14} \approx 77.12\%$. It has a improvement of 25.01% per 256 MB.
   - When 1024 MB is added, $\lfloor (1280 - 96) \div 48 \rfloor = 24$ processes can be store simultaneously in memory, the CPU utilisation is $1 - 0.9^{24} \approx 92.02\%$. It has a improvement of 16.23% per 256 MB.

   Comparing three situations above, we find that adding the first 256 MB is the most beneficial and would be worth the investment.

Problem 2. Keymap in MINIX 3

In order to finish building key mapping, we first need to add one hook `SF7` in `struct hook_entry` which contains in `"dmp.c"` file under `/usr/src/servers/is` directory.

```
struct hook_entry {
    int key;
    void (*function)(void);
    char *name;
} hooks[] = {
    { F1,   proctab_dmp, "Kernel process table" },
    { F3,   image_dmp, "System image" },
    { F4,   privileges_dmp, "Process privileges" },
    { F5,   monparams_dmp, "Boot monitor parameters" },
    { F6,   irqtab_dmp, "IRQ hooks and policies" },
    { F7,   kmessages_dmp, "Kernel messages" },
    { F8,   vm_dmp, "VM status and process maps" },
    { F10,  kenv_dmp, "Kernel parameters" },
    { SF1,  mproc_dmp, "Process manager process table" },
    { SF2,  sigaction_dmp, "Signals" },
    { SF3,  fproc_dmp, "Filesystem process table" },
    { SF4,  dtab_dmp, "Device/Driver mapping" },
    { SF5,  mapping_dmp, "Print key mappings" },
    { SF6,  rproc_dmp, "Reincarnation server process table" },
    { SF7,  pcsn_dmp, "Display the number of currently running processes" },
    { SF8,  data_store_dmp, "Data store contents" },
    { SF9,  procstack_dmp, "Processes with stack traces" },
};
```

Next, we will modify `"dmp_kernel.c"` (`/usr/src/servers/is/dmp_kernel.c`) for our implementation of pcsn_dmp funtion. At the front line, I add `../pm/mproc.h"`.

```
void pcsn_dmp()
{
  struct mproc *mp;
  int i,n=0;
  if(getsysinfo(PM_PROC_NR, SI_PROC_TAB, mproc, sizeof(mproc))!=OK){
    printf("Error obtaining table from PM. Perhaps recompile IS?\n");
    return;
  }
  for(i=0;i<NR_PROCS;i++){
    mp=&mproc[i];
    if(mp->mp_pid==0 && i!=PM_PROC_NR) continue;
    n++;
  }
  printf("The number of the currently running process is: %d\n",n);
}
```

The last file is `minix/servers/is/proto.h` where I add declaration of the function.

```
void pcsn_dmp(void);
void proctab_dmp(void);
void procstack_dmp(void);
void privileges_dmp(void);
void image_dmp(void);
void irqtab_dmp(void);
void kmessages_dmp(void);
void monparams_dmp(void);
void kenv_dmp(void);
```

Finally, I build the MINIX kernal and typing *SHIFT+F7*.

```
cd /usr/src
make build
reboot
```

```
        Key   Description
------------------------------------------------------------------------------
         F1.  Kernel process table
         F3.  System image
         F4.  Process privileges
         F5.  Boot monitor parameters
         F6.  IRQ hooks and policies
         F7.  Kernel messages
         F8.  VM status and process maps
        F10.  Kernel parameters
   Shift+F1.  Process manager process table
   Shift+F2.  Signals
   Shift+F3.  Filesystem process table
   Shift+F4.  Device/Driver mapping
   Shift+F5.  Print key mappings
   Shift+F6.  Reincarnation server process table
   Shift+F7.  Display how many processes are running
   Shift+F8.  Data store contents
   Shift+F9.  Processes with stack traces

The number of the currently running process is 41
```

Figure 2: Key Mapping Complete