Pseudo Code Sheet - Operting Systems - 24296

1 Arrays and Strings

```
string str = "hello";
string str_aux;
str_aux = str;      // Copy string
str == str          // Comparison
```

```
char* str = "hello";
char str_aux[80];
strcpy(str_aux,str)
strcmp(str,str) == 0
```

2 I/O: Read, Write, Open, Close

```
int open(name);
int read(fd, buf, nbyte);
int write(fd, buf, nbyte);
int close(fd);

lseek(int fd, int bytes)
```

```
int open(char* name, O_CREAT | O_RDWR, 0644);
int read(int fd, void *buf, int nbyte);
int write(int fd, void *buf, int nbyte);
int close(int fd);

lseek(int fd, int bytes, SEEK_SET)
```

3 Processes: Fork, Wait, Exec, Exit

```
int fork();
int wait();
exec(name);
exit();
int getpid();
```

```
int fork();
int wait(int* status);
execvp(char *name, char* argv[]);
exit(int status);
int getpid();
```

4 Threads: Create, Join and Synchronization

```
thread_create(function,params)
thread_join()

fthread(params) { }

thread_lock lock;
mutex_lock(lock)
mutex_unlock(lock)

condition  cond;
cond_wait(cond, lock);
cond_signal(cond);


semaphore sem;
sem_ini(sem,value)
sem_wait(sem)
sem_signal(sem)


int file_lock(fd, start, len);
file_unlock(fd, start, len);
```

```
int pthread_create(pthread_t *thread, NULL, function, params);
int pthread_join(pthread_t thread, NULL);

void* fthread(void* params) { return NULL; }

pthread_mutex_t lock;
pthread_mutex_lock(&lock);
pthread_mutex_unlock(&lock);

pthread_cond_t  cond;
pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t* mutex);
pthread_cond_signal(pthread_cond_t *cond);
```

5 Sockets

```


sockfd = socket();

bind(sockfd, serv_addr);
server_listen(sockfd);
newsockfd = accept(sockfd);

client_connect(sockfd, serv_addr);
```

```
struct sockaddr serv_addr;

sockfd = socket(AF_INET, SOCK_STREAM, 0);
fill_addr(NULL, argv[1], &serv_addr);
bind(sockfd, &serv_addr, sizeof(serv_addr));
listen(sockfd,100);
newsockfd = accept(sockfd, &cli_addr,  &clilen);

connect(sockfd, &serv_addr, sizeof(serv_addr));
```

6 Shared Memory

```
int data = shm_get(shmkey);
```

```
shmid = shmget(shmkey, SHM_SIZE, 0666 | IPC_CREAT);
int *shmpointer = shmat(shmid, NULL, 0);
```