Report

Problem 1:

1. Write a c program in user named test_problem_1 which calls the syscall echo_simple with the argument string as parameter.
2. Add the user program in the Makefile
3. Add the stubs for the syscall in usys.pl and in user.h
4. Add the syscall number in syscall.h
5. Add the syscall handler in syscall.c
6. Add the syscall function in sysproc.c

Problem 2:

1. Write a c program in user named test_problem_2 which calls the syscall echo_kernel with the argument array of string as parameter.
2. Add the user program in the Makefile
3. Add the stubs for the syscall in usys.pl and also in user.h
4. Add the syscall number in syscall.h
5. Add the syscall handler in syscall.c
6. Add the syscall function in sysproc.c by referring the exec syscall

---

Problem 3:

1. Copy the provided program into user directory
2. Add the stubs for the syscall in usys.pl and also in user.h
3. Add the syscall number in syscall.h
4. Add the syscall handlers in syscall.c also add the array for syscall names named as "suscallnames"
5. Add mask array to store the binary and traced int to have record

6. Add the syscall function in sysproc.c which coverts the given arg to binary and stores in mask.
7. set traced to 1
8. set traced to 0 in syscall "sbrk"

Problem 4:

1. No Lead

Problem 5:

1. Copy the provided header file into kernel directory
2. Write a c program in user named test_problem_5 which calls the syscall get_process_info
3. Add the stubs for the syscall in usys.pl and also in user.h
4. Add the syscall number in syscall.h
5. Add the syscall handlers in syscall.c
6. Added the syscall function in sysproc.c (I was getting some error related to dereferencing of pointer so i commented the code out)