Problem 1.

ecx	0x3000
edx	0x9000
eip	0x4000

0x9000	
0x9004	0x2222
0x9008	0x5000
0x900c	0x4000
0x9010	0x8888
0x9014	0x6000

Problem 2.

a.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

struct var{
char x[100];
char* y;
};

void foo(char *argv[])
{
  struct var stu;
  strcpy(stu.x, argv[1]);
  strcpy(stu.y, argv[2]);
}

int main(int argc, char *argv[])
{
  if (argc != 3)
    {
      fprintf(stderr, "target1: argc != 3\n");
```

```
    exit(EXIT_FAILURE);
   }
  setuid(0);
  foo(argv);
  return 0;
}
```

b.
Yes. One way is to write shellcode in the heap and redirect the instruction pointer to the heap. Another way is to use ROP to execute instructions in libc.so

Problem 3.
First, we can set hdr->nlen=8192, then nlen=8192. Then 8192 - (nlen+1) will be a very big integer because nlen is an unsigned integer. So we can set hdr->vlen almost arbitrarily and overwrite buf.

Problem 4.
Because of the s flag, anyone can execute this program with the privilege of root. So users in the laura group can write setuid(0); execve("/bin/sh"); to the file and execute the program to get the root.

Defense: if the setuid bit flag is set, the file should only be writable to the owner especially when the owner is root.

Problem 5.
The advantage is that each UID is isolated. So one app cannot influence other apps. If one app is hijacked by an attacker, he can at most control the resources of this app rather than the whole OS.

Problem 6.

a. Change the ownerships of /etc/shadow and /user/bin/passwd to passwd, retain the setuid flag of /user/bin/passwd.
b. The attacker can get the root privilege because of the setuid bit.
c. No, because anyone running the passwd utility has the owner's permission.

Problem 7.

a. During the sleeping time, the attacker can create a link file linking ./file.dat to some system file. Then printing to this file could lead to unexpected behaviors.
b. Still possible if the attacker can insert processes between these two lines.
c. Use O_CREAT and O_EXCL flags in the open function, which will fail if the file exists

Problem 8.

Because seteuid only changes the effective user id, the attacker can turn the effective user id back to root by setuid(0) afterwards. So the attacker can exploit the bug and add setuid(0) preceding to his shellcode.