3 problems. 40 points. 25 minutes *Closed book. Closed notes. No electronic device.* *Write your name above.*
Assume the Ubuntu environment when answering these questions.

1. [20 points]

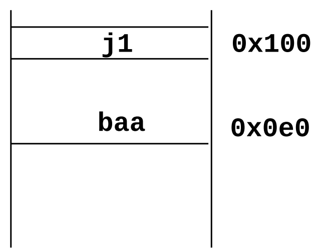
```
int func(int arg)
{
    long j2;
    char buf[16];
    FILE *badfile;
s2:
    badfile = fopen("badfile", "r");
    fread(buf, sizeof(char), 40, badfile);
    return arg+1;
}

int main(int argc, char **argv)
{
    long j1;
    char baa[32];
    j1 = 4;
s1:
    func(j1);
    return 1;
}
```

This program is compiled without Stack Guard, and executed such that variable `j1` is allocated at address `0x00000100`.

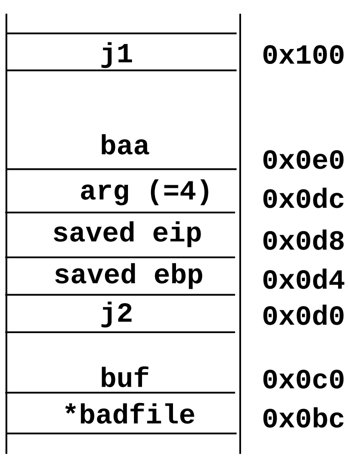
Below, "draw the stack layout" means indicate, in the provided drawing, the contents of the stack from address `0x100` to the top of stack, and give the addresses of the contents at the side.
As usual, grow the stack downward.

(a) Draw the stack layout when control comes to `s1` (i.e., the next instruction to be executed is `func(j1)`).



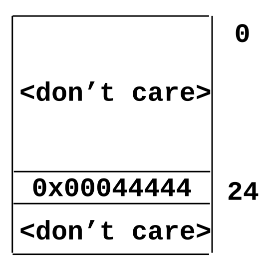
- 4 pts total.
- −1 for incorrect address (eg, decimal instead of hex)
- −1 for having addresses increasing
- −2 for extraneous values
- 0 if no `baa`

(b) Draw the stack layout when control comes to `s2`.



- 14 pts total
- 2 pts for each entry (content, address, order).
- −1 for giving offsets but not addresses.
- −1 for having addresses increasing
- lose (more) points for (more) extraneous entries.

(c) Supply the contents of file `badfile` so that when control returns from `func()`, it starts executing at address `0x00044444`.



- 2 pts total
- 1 pt for address
- 1 pt for content

2. [10 points] Here are two files owned by root:

- `/passwd.txt`: text file that contains user passwords. Root has read-write-execute access. All other users have no access.
- `/chpwd`: executable file that users can run to change their passwords. Root has read-write-execute access. All other users have write-execute access. The setuid bit is set (so it is a set-root-uid file).

Does this configuration allow an ordinary (i.e., non-root) user to delete `passwd.txt`?
If no, explain briefly.
If yes, briefly give the steps of the attack.

Solution

Yes.

- User develops a program, say `xx`, that deletes file `passwd.txt`, e.g.,
  - executable of C program along the following lines:
    ```
    main() {
      remove(/passwd.txt)
    }
    ```
  - bash script along the following lines:
    ```
    #! /bin/bash
    rm -f /passwd.txt
    ```
  - shellcode (to get a root shell from which `passwd.txt` can be deleted).
- User writes `xx` into `/chpwd` (can do this because it has write access).
- User executes `xx`. This deletes `/passwd.txt` because it executes with root privilege.

Grading

- Max 3 points for attempting a buffer-overflow attack on original `chpwd`.
  (You cannot assume that `chpwd` has such a vulnerability.)
- Max 3 points for some explanation as to why attack is not doable.

3. [10 points] For each of the following CPU instructions, indicate whether or not attempting to execute the instruction in user mode results in an (illegal instruction) exception.

Write "EX to indicate that it does result in an exception. Write "NX" otherwise.

- set kernel mode
  Solution: EX
- add the contents of ebx to eax
  Solution: NX
- int 0x80 // software interrupt 0x80
  Solution: NX (does not cause an illegal instruction exception)
- push ebx
  Solution: NX
- disable interrupts
  Solution: EX

Grading: 2 pts for each.