Question 1

I gave you a blank question 1 so that you could have a scratch "essay" box. It was worth 0 points.

Question 2

Our machine has four-byte pointers and is little-endian. The syntax of memcpy is:

```
void memcpy(void *dest, const void *src, int bytes);
```

Here's a procedure:

```
void a(unsigned int *x)
{
  unsigned char *c;
  unsigned int y, z;

  c = (unsigned char *) x;
  y = 0;
  z = 0;

  memcpy(&y, c+2, sizeof(unsigned int));
  memcpy(&z, x+3, 3);

  printf("1 0x%02x\n", c[0]);
  printf("2 0x%02x\n", c[5]);
  printf("3 0x%02x\n", c[10]);
  printf("4 0x%02x\n", c[15]);
  printf("5 0x%08x\n", (unsigned int) (x+1));
  printf("6 0x%08x\n", x[1]);
  printf("7 0x%08x\n", y);
  printf("8 0x%08x\n", z);
}
```

When we run it, x is 0x04e05030, and here are the values of the 16 bytes of memory starting with x:

```
 Address     Value
0x04e05030 0x893b6165
0x04e05034 0x50f6dc88
0x04e05038 0x04e05034
0x04e0503c 0xd4cfa7ce
```

Please answer with the 8 lines of output of this procedure. Please don't put anything extraneous into your answer box.

Question 3

Here are the prototypes to strcpy and strcat:

```
char *strcpy(char *dest, const char *source);
char *strcat(char *dest, const char *source);
```

Here is a procedure:

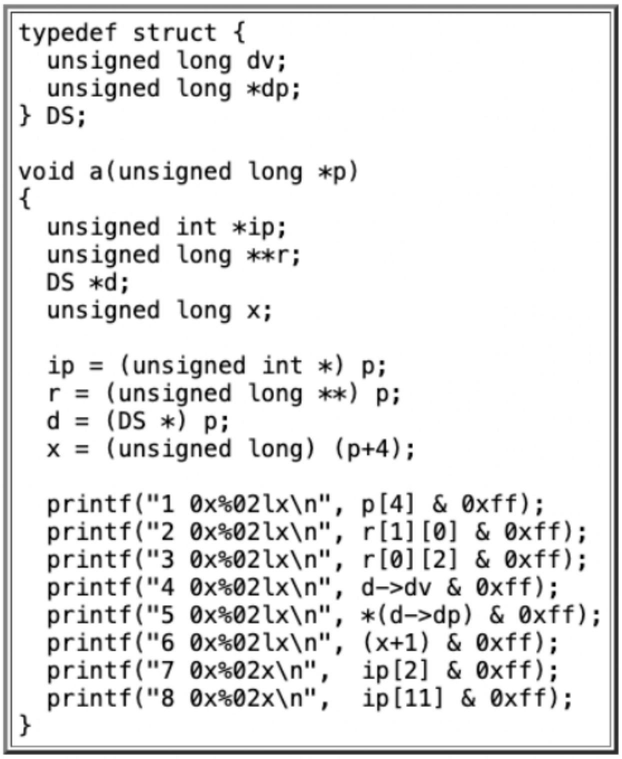
```
void a(char *b, char *c, char *d)
{
  strcpy(b+2, c);
  strcpy(b+12, c);
  strcat(b, d);
  b[2] = '\0';
  printf("1 %s\n", b);
  printf("2 %s\n", b+5);
  printf("3 %s\n", b+10);
  printf("4 %s\n", b+15);
  printf("5 %s\n", b+20);
}
```

Please put what the output to a() is when you run it with the following arguments. I've put the numbers in to help you count:

```
     0123456789012345678901
b = "TennesseeVolunteersWin"
c = "Fred"
d = "123"
```

Question 4

You are on a little-endian machine with 8-byte pointers. You are running the following procedure:



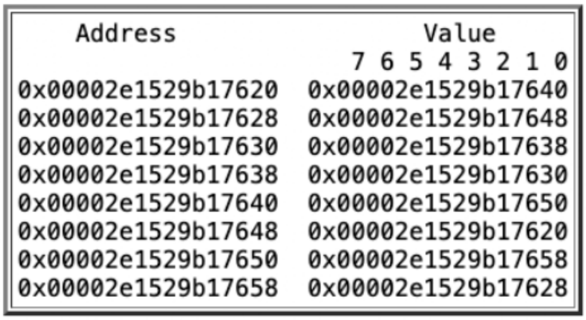
```
typedef struct {
  unsigned long dv;
  unsigned long *dp;
} DS;

void a(unsigned long *p)
{
  unsigned int *ip;
  unsigned long **r;
  DS *d;
  unsigned long x;

  ip = (unsigned int *) p;
  r = (unsigned long **) p;
  d = (DS *) p;
  x = (unsigned long) (p+4);

  printf("1 0x%02lx\n", p[4] & 0xff);
  printf("2 0x%02lx\n", r[1][0] & 0xff);
  printf("3 0x%02lx\n", r[0][2] & 0xff);
  printf("4 0x%02lx\n", d->dv & 0xff);
  printf("5 0x%02lx\n", *(d->dp) & 0xff);
  printf("6 0x%02lx\n", (x+1) & 0xff);
  printf("7 0x%02x\n",  ip[2] & 0xff);
  printf("8 0x%02x\n",  ip[11] & 0xff);
}
```

You run a() with p = 0x00002e1529b17620. Here's the state of memory starting at that address:



```
    Address                Value
                      7 6 5 4 3 2 1 0
0x00002e1529b17620  0x00002e1529b17640
0x00002e1529b17628  0x00002e1529b17648
0x00002e1529b17630  0x00002e1529b17638
0x00002e1529b17638  0x00002e1529b17630
0x00002e1529b17640  0x00002e1529b17650
0x00002e1529b17648  0x00002e1529b17620
0x00002e1529b17650  0x00002e1529b17658
0x00002e1529b17658  0x00002e1529b17628
```

Please enter the output of a().

Question 5

Suppose you are writing a program to test whether two files are hard links to each other.Â Tell me in English how your program would perform that test.

Question 6

Here's program 1:

```
int main()
{
  unsigned int hash, l;
  hash = 0;

  while (fread(&l, sizeof(unsigned int), 1, stdin) == 1) hash = add_to_hash(hash, l);
  printf("0x%x\n", hash);
}
```

And here's program 2:

```
int main()
{
  unsigned int hash, l;
  hash = 0;

  while (read(0, &l, sizeof(unsigned int)) == 1) hash = add_to_hash(hash, l);
  printf("0x%x\n", hash);
}
```

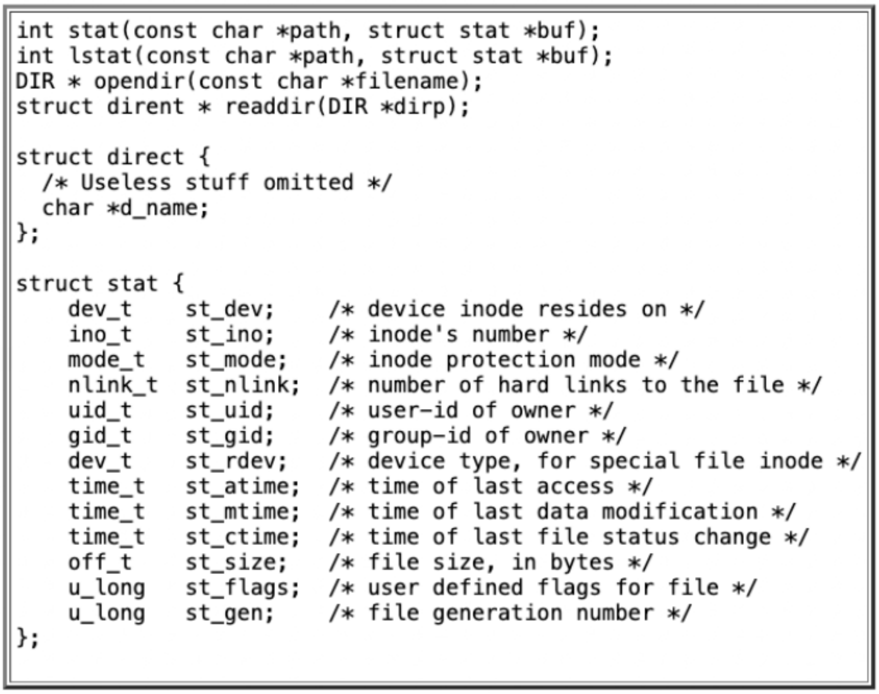
Let's suppose that standard input is four megabytes. Please tell me which of these programs will run faster, and then explain why. Your explanation should be a paragraph, and you should use numbers to support your reasoning. In particular, it will be a good idea for you to estimate how long each program will take to run.

Question 7

Write a program in C to count the number of subdirectories of the current directory that contain a file whose name is **f1.txt**, and whose size is at least 100 bytes. I don't care about include files, so don't bother with them.

If you encounter errors, simply **exit(1)**.

Here are helpful prototypes and structs: Obviously, these are the calls that you should make to solve this problem -- please don't get cute and try popen() or system().



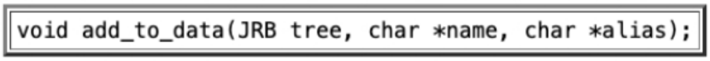
```
int stat(const char *path, struct stat *buf);
int lstat(const char *path, struct stat *buf);
DIR * opendir(const char *filename);
struct dirent * readdir(DIR *dirp);

struct direct {
  /* Useless stuff omitted */
  char *d_name;
};

struct stat {
    dev_t    st_dev;    /* device inode resides on */
    ino_t    st_ino;    /* inode's number */
    mode_t   st_mode;   /* inode protection mode */
    nlink_t  st_nlink;  /* number of hard links to the file */
    uid_t    st_uid;    /* user-id of owner */
    gid_t    st_gid;    /* group-id of owner */
    dev_t    st_rdev;   /* device type, for special file inode */
    time_t   st_atime;  /* time of last access */
    time_t   st_mtime;  /* time of last data modification */
    time_t   st_ctime;  /* time of last file status change */
    off_t    st_size;   /* file size, in bytes */
    u_long   st_flags;  /* user defined flags for file */
    u_long   st_gen;    /* file generation number */
};
```

Question 8

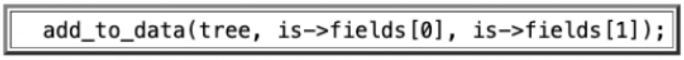
Please implement the procedure add_to_data(), which has the following prototype:



```
void add_to_data(JRB tree, char *name, char *alias);
```

The tree is keyed on name. Each node's val is a dllist of aliases (which are strings). add_to_data() should create a node for the given name if it doesn't exist. Then, it should add the alias to the name's dllist.

It will be called as follows from the main():



```
  add_to_data(tree, is->fields[0], is->fields[1]);
```

Do not bother with include files. I don't care. However, the rest of your code should be polished. You don't have to error check.

Here is relevant information from dllist.h and jrb.h:

```
typedef struct dllist {
  struct dllist *flink;
  struct dllist *blink;
  Jval val;
} *Dllist;

Dllist new_dllist();
void free_dllist(Dllist);
void dll_append(Dllist, Jval);
void dll_prepend(Dllist, Jval);
void dll_insert_b(Dllist, Jval);
void dll_insert_a(Dllist, Jval);
void dll_delete_node(Dllist);
#define dll_traverse(ptr, list) for (ptr = list->flink; ptr != list; ptr = ptr->flink)

typedef struct jrb_node {
  struct jrb_node *flink;
  struct jrb_node *blink;
  Jval key;
  Jval val;
  /* Other stuff */
} *JRB;

JRB make_jrb();
JRB jrb_insert_str(JRB tree, char *key, Jval val);
JRB jrb_find_str(JRB root, char *key);
JRB jrb_find_gte_str(JRB root, char *key, int *found);
void jrb_delete_node(JRB node);
void extern void jrb_free_tree(JRB root);
#define jrb_traverse(ptr, lst) for (ptr = list->flink; ptr != list; ptr = ptr->flink)
```

Question 9

Now, write the following procedure:

```
void print(JRB tree);
```

This should print the tree from the previous question.Â The format should be one line per node of the tree.Â For each node, print the name, followed by a colon, and then the aliases, each separated by a space.