CS360 Midterm 2 - March 21, 2017 - James S. Plank

Put all answers on the answer sheet. In all of these questions, please assume the following:

- Pointers and longs are 4 bytes.
- The machine is little endian, but that doesn't matter in any of these questions.
- There are no segmentation violations or bus errors in any of this code.
- Any assembly code questions use the **jassem** assembly code.
- Any assembly code that corresponds to compiled C code is unoptimized.

Question 1

Tell me the output of the program below:

```
int main()
{
  unsigned int x;
  unsigned char c;
  double *pdl, *pdh;
  unsigned char *pcl, *pch;

  x = 0x84b3a03e;
  c = (unsigned char) x;

  printf("1: 0x%08x\n", (x >> 8));
  printf("2: 0x%08x\n", (x << 8));
  x = x ^ c;
  printf("3: 0x%08x\n", x);

  printf("4: 0x%08x\n", (c | 0xf0));
  printf("5: 0x%08x\n", (c & 0xf0));

  c = (c << 4);
  printf("6: 0x%08x\n", c);

  pcl = &c;
  pdl = (double *) pcl;
  pdh = &(pdl[3]);
  pch = (unsigned char *) pdh;

  printf("7: %ld\n", (pch - pcl));
  printf("8: %ld\n", (pdh - pdl));

  exit(0);
}
```

Question 2

You've received a coveted interview at the tech giant Grapple. You goal is to work in their "upgrade" department, where you and your friends will decide which features of their products you'll change from upgrade to upgrade without telling anyone. It's the best job! But to get there, you have to work in their Systems Programming department.

At your interview, they give you a laptop, and tell you that there is as least one non-directory file on the system, that has exactly six hard links. Your job is to find one of these files, find two of the 6+ links to it, and print the two links' full path names.

Now, I won't make you write this program, but I'm going to give you a skeleton and then ask you some questions. The skeleton is on the next page.

- **Question A**: What is **d** for, and what system/library call to you use to initialize it?
- **Question B**: What is **de** for, and what system/library call to you use with it?
- **Question C**: What is **buf** for, and what system/library call to you use with it?
- **Question D**: Which fields of **buf** do you use and why?
- **Question E**: Your program will call **malloc()** and **free()**. On which variable, and why?
- **Question F**: Your program will also call **strdup()**. On which variable and why?

Question 3

In one paragraph, explain to me why it is that **Program A** to the right is fast and **Program B** is slow, even though they produce the same output on the same input. Give me details in your explanation -- don't just use buzz-words.

```
int main()                                          /* Program A */
{
  while (fread(&c, 1,1, stdin) == 1) fwrite(&c, 1, 1, stdout);
  exit(0);
}
```

```
int main()                                          /* Program B */
{
  while (read(0, &c, 1) == 1) write(1, &c, 1);
  exit(0);
}
```

Question 2 - The skeleton of code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/stat.h>

void find_6(char *dir, char **p1, ino_t *number)
{
  DIR *d;
  struct dirent *de;
  struct stat buf;
  char *pn;
  int length;

  /* Initialization ............... */
  for (........) {
    .........
    if (..........) {
      if (strcmp(de->d_name, ".") != 0 &&
          strcmp(de->d_name, "..") != 0) {
        find_6(pn, p1, number);
      }
    } else {
      if (.........) {
        if (*p1 == NULL) {
          *p1 = ....;
          *number = ....;
        } else if (.......) {
          printf("%s\n%s\n", *p1, pn);
          exit(0);
        }
      }
    }
  }
  /* Final stuff ................. */
}

int main()
{
  ino_t num;
  char *firstpn;

  firstpn = NULL;
  find_6("/", &firstpn, &num);
  return 0;
}
```

Useful Prototypes

```
strcpy(char *to, char *from)
strcat(char *to, char *from)
int strcmp(const char *s1, const char *s2)
int strncmp(const char *s1, const char *s2, int n)
char *strchr(char *s, char tofind)
char *strdup(char *s)
mempcy(char *to, char *from, int nbytes)
void *malloc(size_t size)
void free(void *ptr)
```

More Useful Prototypes

```
int open(char *path, int flags [ , int mode ] )
int close(int fd)
ssize_t read(int fd, void *buf, size_t count)
ssize_t write(int fd, const void *buf, size_t count)
off_t lseek(int fildes, off_t offset, int whence)

FILE *fopen(char *path, char *mode)
int fclose(FILE *f)
int fread(void *ptr, int size, int nbytes, FILE *f)
int fwrite(void *ptr, int size, int nbytes, FILE *f)
int fseek(FILE *stream, long offset, int whence)

int stat(char *path, struct stat buf)
struct stat {
  mode_t st_mode;     /* File mode (see mknod(2)) */
  ino_t st_ino;       /* Inode number */
  nlink_t  st_nlink;  /* # of hard links to the file */
  uid_t    st_uid;    /* user-id of owner */
 /* other stuff omitted */
}
int S_ISDIR(mode_t mode) /* Is the file a directory? */

DIR *opendir(char *path)
int closedir(DIR *d)
struct dirent *readdir(DIR *D)
struct dirent {
  char *de_name;
  /* other stuff omitted */
}

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);
int dll_empty(Dllist);

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

JRB make_jrb();
JRB tree, char *key, Jval val);
JRB jrb_insert_str(JRB tree, char *key, Jval val);
JRB jrb_insert_int(JRB tree, int ikey, Jval val);
JRB jrb_insert_dbl(JRB tree, double dkey, Jval val);
JRB jrb_find_str(JRB root, char *key);
JRB jrb_find_int(JRB root, int ikey);
JRB jrb_find_dbl(JRB root, double dkey);
void jrb_delete_node(JRB node);
void jrb_free_tree(JRB root);
```

CS360 Midterm 2 - March 21, 2017 - James S. Plank - Page 3

Question 4

Please write the assembly code for the following:

```
int b(int *c)
{
  return c[3];
}

void f(int i)
{
  int g[4];

  g[i] = 10;
}

void d(int **e, int i)
{
  e[i][3] = 5;
}
```

Question 5

Please tell me the first four instructions, and the last five instructions of the procedure below:

```
int b(int i, int j)
{
  int k, l;

  k = f(5);
  k = c(e(i)+e(j));
  l = f(k);
  return 3;
}
```

Question 6

The following procedure, when translated into assembly, takes 13 instructions. These start at memory location 0x1450.

```
int k(int i, int j)
{
  int n, m;

  n = i+3;
  m = j+2;
  return n+m;
}
```

The following procedure, when translated into assembly, takes 12 instructions. These start at memory location 0x1600. The **jsr** statement is in memory location 0x161c.

```
int h(int i, int j)
{
  int n;

  n = k(j+5, i);
  return n;
}
```

Question 6, Continued

We are executing a program that includes these procedures, and `h(20,30)` has been called. This, of course, calls `k(35,20)`. Suppose we are examining the state of memory and registers just before the `ret` instruction of `k()` executes. The value of the frame pointer is `0x7ff800`. On the answer sheet, I have boxes for the registers **r0**, **pc**, **sp** and **fp**. Please fill in their values. These values should be numbers, in either decimal or hexadecimal -- precede the hex numbers with "0x".

I also have two sets of boxes for memory locations `0x7ff7f0` through `0x7ff824`. In the first set of boxes, label the memory locations. An example label would be "variable n in k()", which you can shorten to "n in k". If you can't label a location, simply put a dash there.

In the second set of boxes, fill in the values of the memory locations. If you don't know what a value is, put a dash there. These values should be numbers, in either decimal or hexadecimal -- precede the hex numbers with "0x".