CS140 Midterm Exam - October 4, 2007

James S. Plank

Answer all questions. Do so on a separate sheet of paper.

Question 1

Behold the following C program:

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

main(int argc, char **argv)
{
  char *s, *u, *v;

  s = argv[1];
  v = strchr(s, 'G');
  u = strdup(s);

  if (v == NULL) exit(0);

  *v = 'g';
  v += 5;
  *v = 'i';

  printf("0x%x 0x%x\n", s, u);

  printf("argc: %d\n", argc);
  printf("argv[1]: 0x%x %s\n", argv[1], argv[1]);
  printf("s: 0x%x %s\n", s, s);
  printf("u: 0x%x %s\n", u, u);
  printf("v: 0x%x %s\n", v, v);
  printf("strstr: 0x%x %s\n", strstr(u, "ge"), strstr(u, "ge"));
}
```

Part 1: Suppose we compile this into an executable called **pex**, and run it from the shell as below. I give the first line of output. What are the remaining lines of output? Put question marks where you don't know the answer, but answer everything that you can.

```
UNIX> pex Curious-George Prince-George Eddie-George
0xbfffedd8 0x3000f0
```

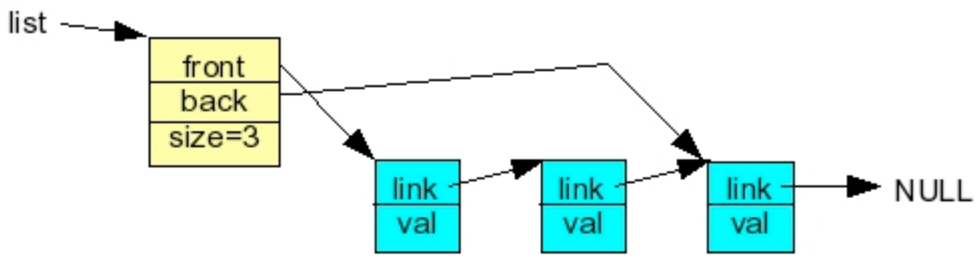
Part 2: You'll note that the above program does no error checking. Give two examples of running it from the command line where the output will be non-deterministic (e.g. a Bus error, Segmentation violation, or some kind of random output) and state why these examples are problemmatic. Note, you don't need to specify what the output is, just why the examples are problemmatic.

Question 2

Recall the following potential implementation/prototypes for a singly-linked list:

```
typedef struct node {
  struct node *link;
  Jval val;
} Node;
```

```
typedef struct {
  Node *front;
  Node *back;
  int size;
} List;
```



- **List *new_list()** - Create and return a new empty list. This will consist solely of the header struct whose **front** and **rear** pointers will be NULL.
- **int list_empty(List *l)** - Is a list empty?
- **Node *list_front(List *l)** - return a pointer to the first node on the list, or NULL if the list is empty.
- **void list_prepend(List *l, Jval v)** - allocate a new node with value **v** and put it on the front of the list.
- **void list_append(List *l, Jval v)** - allocate a new node with value **v** and put it on the end of the list.
- **void list_insert_after(List *l, Node *n, Jval v)** - allocate a new node with value **v** and put it after node **n**.
- **int list_size(List *l)** - Return the number of nodes on the list.
- **void free_list(List *l)** - free up the header structure and the nodes.

Implement the three procedures: **new_list()**, **list_prepend()**. and **free_list()**.

Question 3

Suppose you are designing and implementing a data structure that many people are going to use. How would you use a **(void *)** to perform information hiding?

Question 4

Behold the following program:

```
#include <stdio.h>
#include <stdlib.h>
#include "fields.h"
#include "stack.h"
#include "queue.h"
#include "dllist.h"

main()
{
  IS is;
  Queue q;
  Stack s;
  Dllist l;
  int i;
  Jval j;

  is = new_inputstruct(NULL);

  while (get_line(is) > 0) {
    q = new_queue();
    s = new_stack();
    l = new_dllist();
    for (i = 0; i < strlen(is->fields[0]); i++) {
      queue_enqueue(q, new_jval_c(is->fields[0][i]));
      stack_push(s, new_jval_c(is->fields[0][i]));
      dll_append(l, new_jval_c(is->fields[0][i]));
      dll_prepend(l, new_jval_c(is->fields[0][i]));
    }
    while (!queue_empty(q)) {
      j = queue_dequeue(q); printf("%c", j.c);
      j = stack_pop(s); printf("%c", j.c);
      printf("%c", l->flink->val.c); dll_delete_node(l->flink);
      printf("%c", l->flink->val.c); dll_delete_node(l->flink);
      printf(" ");
    }
    printf("\n");
    free_queue(q);
    free_stack(s);
    free_dllist(l);
  }
}
```

What will the output of this program be when the following file is given as standard input:

```
1 2 3
Fred Luther Dontonio
Eddie Prince Curious

Frank Lloyd Wright
22
```

Prototypes

```
typedef union {
    int i;
    double d;
    void *v;
    char *s;
    char c;
    ...
  } Jval;

Jval new_jval_i(int);
Jval new_jval_d(double);
Jval new_jval_v(/* void */);
Jval new_jval_s(char *);
Jval new_jval_c(char);
```

```
char *strdup(char *s);
void strcpy(char *dest, char *source);
char *strchr(char *s, int ch);
char *strrchr(char *s, int ch);
char *strstr(char *s, char *tofind);
int getchar();
```

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

extern Dllist new_dllist();
extern void free_dllist(Dllist);
extern void dll_append(Dllist, Jval);
extern void dll_prepend(Dllist, Jval);
extern void dll_delete_node(Dllist);
```

```
#define MAXLEN 1001
#define MAXFIELDS 1000

typedef struct inputstruct {
  char *name;
  int line;
  char text1[MAXLEN];
  int NF;
  char *fields[MAXFIELDS];
  ...
} *IS;

IS new_inputstruct(char *name);
int get_line(IS);
void jettison_inputstruct(IS);
```

```
typedef void *Queue;

Queue new_queue();
void queue_enqueue(Queue q, Jval v);
void free_queue(Queue q);
Jval queue_dequeue(Queue q, Jval v);
int queue_empty(Queue q);
```

```
typedef void *Stack;

Stack new_stack();
void stack_push(Stack s, Jval v);
void free_stack(Stack s);
Jval stack_pop(Stack s, Jval v);
int stack_empty(Stack s);
```