Princeton University
COS 217: Introduction to Programming Systems
Spring 2005 Midterm Exam Answers

Question 1

```
struct buffer {
   int size;
   char *text;
   int f;
   int b;
};
```

Question 2

```
#define INVARIANTS(buf)           \
   ( ((buf) != NULL) &&           \
     ((buf)->text != NULL) &&     \
     (0 <= (buf)->f) &&           \
     ((buf)->f <= (buf)->b) &&    \
     ((buf)->b <= (buf)->size) )
```

Question 3

```
void Buffer_forward(Buffer_T buf) {
   assert(INVARIANTS(buf));
   assert(buf->b < buf->size);
   buf->text[buf->f] = buf->text[buf->b];
   (buf->f)++;
   (buf->b)++;
   assert(INVARIANTS(buf));
}
```

Question 4

```
void Buffer_fillFromFile(Buffer_T buf, FILE *file) {
   int c;
   assert(INVARIANTS(buf));
   assert(file != NULL);
   while (c = getc(file); c != EOF; c = getc(file))
      Buffer_insert(buf, (char)c);
   assert(INVARIANTS(buf));
}
```

Alternative:

```
Buffer_T Buffer_newFromFile(FILE *file) {
   Buffer_T buf;
   int c;
   assert(file != NULL);
   buf = Buffer_new();
   while (c = getc(file); c != EOF; c = getc(file))
      Buffer_insert(buf, (char)c);
   assert(INVARIANTS(buf));
   return buf;
}
```

Question 5

```
SymTable_T SymTable_new(void);

void SymTable_free(SymTable_T oSymTable);

unsigned int SymTable_getLength(SymTable_T oSymTable);

int SymTable_put(SymTable_T oSymTable, Buffer_T buf, const void *pvValue);
int SymTable_remove(SymTable_T oSymTable, Buffer_T buf);

int SymTable_contains(SymTable_T oSymTable, Buffer_T buf);

void *SymTable_get(SymTable_T oSymTable, Buffer_T buf);

void SymTable_map(SymTable_T oSymTable,
   void (*pfApply)(Buffer_T buf, void *pvValue, void *pvExtra),
   const void *pvExtra);
```

Question 6

The buffer-Symtable should not own the buffers.   You received full credit if any two of the following reasons were mentioned:

1. It's safe to do it this way, since the client promises not to modify the contents of the buffer while it's in the symbol table. Buffer_forward and Buffer_back should be seen as not modifying the "abstract" contents.

2. It's more efficient to do it this way, because it means that buffers don't have to be copied when they're put in the SymTable. This is more of a big deal than for the "library books" example, because buffers are expected to e really big.

3. If the SymTable makes a copy of the buffer, then it may need access to the representation of the buffer struct, which is bad.

Question 7

```
#define HASH_MULTIPLIER 65599

static unsigned int hash(Buffer_T buf) {
   int i;
   int size;
   unsigned int h = 0;
   size = Buffer_size(buf);
   for (i = 0; i < size; i++)
      h = h * HASH_MULTIPLIER + Buffer_getChar(buf, i);
   return h;
}
```

The whole point of problem 7 -- indeed, the whole point of the entire semester -- is that client functions such as this one should not directly manipulate the representation of the abstract data type. buffer-SymTable is a client of the Buffer module. This issue is bound to come up on the final, and be worth a lot more than 5 points.

If you mentioned Buffer_getchar in your solution to problem 7, then you get a gold star on your exam. Otherwise you get a frowny-face.