NAME:

Login name:

Computer Science 217
First Midterm Test
March 7, 2002
2PM-4PM

This test is 9 questions. Put your name on every page, and write out and sign the Honor Code pledge before turning in the test.

``I pledge my honor that I have not violated the Honor Code during this examination."

Question	Score
1	
2	
3	
4	
5	
6	
7	
8	
9	
Total	

QUESTION 1 *(12 POINTS)*

Assume a machine for which a **`char`** takes 1 byte, an **`int`** takes 4 bytes, a **`float`** takes 4 bytes, any pointer takes 4 bytes, and no padding is added to any structure. Write the output of the following program (enclose your answer in a box).

```
#include <stdio.h>

struct S1 {
    int a;
    int b;
} a, *b;

struct S2 {
    int c;
    int d[16];
} c;

struct S3 {
    int e;
    char *f[16];
} d, *e[2];

int main()
{
    struct S1 f[2];
    static struct S1 g[2];
    char *h[] = { “Hello”, “World” };

    printf(“a) %d\n”, sizeof(a));
    printf(“b) %d\n”, sizeof(b));
    printf(“c) %d\n”, sizeof(c));
    printf(“d) %d\n”, sizeof(d));
    printf(“e) %d\n”, sizeof(e));
    printf(“f) %d\n”, sizeof(f));
    printf(“g) %d\n”, sizeof(g));
    printf(“h) %d\n”, sizeof(h));
    printf(“i) %d\n”, sizeof(h[1]));
    printf(“j) %d\n”, sizeof(h[1][1]));
    printf(“k) %d\n”, sizeof(f+1));
    printf(“l) %d\n”, sizeof(*g));
}
```

QUESTION 2 *(8 POINTS)*

a) Describe the most important property of “opaque pointers”:
(One short sentence)

b) What feature of the C programming language ensures that this property is achieved?
(One or two short sentences)

QUESTION 3 *(8 POINTS)*

a) What is the main difference between a declaration and a definition?
(One or two short sentences)

b) Why does the C programming language have declarations? Why not just definitions?
(One or two short sentences)

QUESTION 4 *(12 POINTS)*

For each of the following code fragments, determine whether or not it is likely to cause a run-time program error (assume that the code compiles without warnings or errors). If there is a run-time program error, please write “ERROR” next to it and describe the problem in one or two short sentences. Otherwise, simply write “OK” next to it.

```
a) void *pVoid = malloc(8 * sizeof(int));
   int *pInt = pVoid;
   free(pInt);
```

```
b) float a[] = { 1.0, 0.0, 1.0, 0.0 };
   float *b = &a[2];
   float *c = b-- + 1;
   float result = *b / *c;
```

```
c) void *my_alloc(void *ptr, int size)
   {
       if (ptr) return realloc(ptr, size);
       else return malloc(size);
   }

   void my_free(void *ptr)
   {
       free(ptr);
       ptr = 0;
   }

   int main()
   {
     void *ptr = my_alloc(0, 4);
     my_free(ptr);

     ptr = my_alloc(ptr, 4);
     my_free(ptr);

     return 0;
   }
```

QUESTION 5 *(12 POINTS)*

Consider the following C program. For each identifier listed in the box below, indicate from which area of the virtual address space the system allocates storage. Fill your answers in the spaces provided in the box below by writing to the right of each identifier one of the following words: “stack,” “heap,” or “global” (where global = data or bss).

```
1     #include <stdio.h>
2     #include <string.h>
3
4     char *name = 0;
5
6     int ParseArguments(int argc, char **argv)
7     {
8       char ext [] = { '.', 'd', 'a', 't', '\0' };
9       int found = 0;
10
11      argc--; argv++;
12      while (argc > 0) {
13        if (!strcmp(*argv, "-name")) {
14          argv++; argc--;
15          name = malloc(strlen(*argv) + sizeof(ext) + 1);
16          strcpy(name, *argv);
17          found = 1;
18        }
19        else if (!strcmp(*argv, "-ext")) {
20          argv++; argc--;
21          strncpy(ext, *argv, sizeof(ext));
22        }
23        else {
24          printf("Bad arg\n");
25          return 0;
26        }
27        argv++; argc--;
28      }
29
30      if (found)
31        strcat(name, ext);
32
33      return found;
34    }
```

Line Number	**Identifier Name**	**Allocated from Region:**
4	name	______________
6	argc	______________
6	argv	______________
6	*argv	______________
6	**argv	______________
8	ext	______________
8	'd'	______________
9	found	______________
13	"-name"	______________
15	*name	______________
21	*ext	______________
21	sizeof(ext)	______________

QUESTION 6 *(12 POINTS)*

For each of the following code fragments, determine whether or not it is likely to cause a run-time program error (assume that the code compiles without warnings or errors). If there is a run-time program error, please write “ERROR” next to it and describe the problem in one or two short sentences. Otherwise, simply write “OK” next to it.

```
a) #include <stdio.h>
   #include <string.h>

   char *MakeFilename(char *name, char *ext)
   {
       char buffer[BUFFERSIZE];     <- Assume this is big enough
       strcpy(buffer, name);
       strcat(buffer, ext);
       return buffer;
   }

   int main(int argc, char **argv)
   {
       if (argc > 2) {
           char *filename = MakeFilename(*argv[1], *argv[2]);
           printf(“%s\n”, filename);
       }
   }
```

```
b) #include <stdio.h>

   int main(int argc, char **argv)
   {
       if (argc > 1) {
           int n = atoi(argv[1]);
           if (n = 0) printf(“Infinity\n”);
           else printf(“%f\n”, 1.0 / n);
       }
   }
```

```
c) #include <stdlib.h>
   #include <stdio.h>

   int main()
   {
       int *ip = calloc(4, sizeof(int));
       float *fp = calloc(4, sizeof(float));
       printf(“%f\n”, *ip + *fp);
   }
```

QUESTION 7 (12 POINTS)

Suppose you are given the following declarations and definitions for implementing a *sorted* doubly-linked list. As in the example from precept, you can assume that the **psMarkerNode** field of the **List** points to a marker node that was created with **pvItem** set to NULL, and with **psPrevNode and psNextNode** fields set to itself when the list was initialized. For each node, the **pvItem** field points to the client's data, the **psNextNode** field points to the next node on the list or the marker node if the list is empty, and the **psPrevNode** field points to the previous node on the list or the marker node if the list is empty.

Your task is to write a function "**int List_remove(List_T oList, void *pvData)**" that removes from **oList** the node whose **pvItem** field matches the function's **pvData** parameter according to the list's **pfCompare** function (which returns a negative, zero, or positive integer when the first data is less, equal, or greater than the second). Your function should return 1 if a matching node was found and removed, and it should return 0 otherwise. Your answer will be graded based on style, robustness, and correctness (you can omit comments, if you like).

```
typedef struct List *List_T;

struct ListNode {
    void *pvItem;
    struct ListNode *psPrevNode, *psNextNode;
};

struct List {
    struct ListNode *psMarkerNode;
    int (*pfCompare)(const void *pvData1, const void *pvData2);
};

int List_remove(List_T oList, void *pvData)
{
    <add your code here>

}
```

QUESTION 8 (12 POINTS)

Write code that implements the following two functions of a string allocation module:

```
      char *String_malloc(int iSize);
      void String_free(char *pcString);
```

The key feature of your implementation should be that it checks whether or not a pointer passed to **`String_free`** was actually allocated by a previous call to **`String_malloc`** (this will help find memory deallocation bugs at run-time). You can achieve this goal by storing a special value of your choosing in the byte just before any block of memory returned by **`String_malloc`** and checking whether that special value is present in any memory block passed to **`String_free`**. You can use the standard **`malloc`** and **`free`** in your implementation to allocate memory from the heap. Your answer will be graded based on style, robustness, and correctness (you can omit comments, if you like).

QUESTION 9 (12 POINTS)

A subset of the interface for the Set ADT of assignment 2 appears below. Describe a set of tests that verify the correctness of an implementation of this subset. You do not have to provide code, just a precise and detailed description of your ideas regarding what would be an effective testing strategy. You will be graded based on how well your description identifies and attempts to satisfy the properties of a good test program outlined in lecture (note: the test program we provided for assignment 2 does not meet many of these properties).

```
void        Set clear(Set T oSet);
int         Set_getLength(Set_T oSet);
int         Set_put(Set_T oSet, const void *pvKey, void *pvValue);
int         Set_remove(Set_T oSet, const void *pvKey);
const void *Set_getKey(Set_T oSet, const void *pvKey);
void       *Set_getValue(Set_T oSet, const void *pvKey);
```