Exam Redux

These questions should look familiar to most of you. We'll talk a bit about these to help remind you of where you've been!

b) The following code snippet has exactly one error that will cause a problem compiling or executing. Identify the error and indicate whether it is detected during compilation or during execution.

```
main()
{
   string a, b;

   a = Concat("CS", "106A");
   b = "Rocks!";
   strcat(a, b);
}
```

b2) (same instructions as above)

```
main()
{
   int *binky[5];

   binky[4][4] = 8;
}
```

c) You are given the following type declaration:

```
typedef struct {
    int x;
    int y;
} *pointT;
```

Write the **`CreatePoint`** function which given x and y values will return a point initialized to those values:

```
pointT CreatePoint(int x, int y)
```

f) The following program is supposed to print the minimum and maximum values from an integer array. The program compiles but does not execute correctly. Identify the flaw and indicate how you would fix it.

```
void GetMinAndMax(int arr[], int n, int min, int max)
{
   int i;

   if (n <= 0) Error("No values in array!");
   min = max = arr[0];
   for (i = 1; i < n; i++) {
      if (arr[i] < min)
          min = arr[i];
      else if (arr[i] > max)
          max = arr[i];
   }
}

main()
{
   int arr[5] = {4, 17, 10, 15, -6};
   int min, max;

   GetMinAndMax(arr, 5, min, max);
   printf("Min %d Max %d\n", min, max);
}
```

Problem 3: Pointers and program tracing (20 points)
As you did for the first problem in the pointer problem set, trace through the following program and draw a memory diagram showing the state of memory at the indicated point. Draw each variable as a box containing its current value and label each with the variable name. Uninitialized memory locations should be left blank. Clearly distinguish between stack and heap memory.

```
typedef struct {
   int pages;
   string title;
} bookT;

main()
{
   int i;
   bookT *desk[2];
   bookT *library;
   bookT shelf[2];

   shelf[1].pages = 100;
   library = NewArray(3, bookT);
   for (i = 0; i < 3; i++) {
       library[i].pages = 5 + 4 * i;
       library[i].title = Concat("Big", "Book");
   }
   desk[0] = &shelf[0];
   desk[1] = library;
   *desk[0] = library[1];
   library = New(bookT *);
   library->pages = StringLength(shelf[0].title);
   Read(desk[1], *library, &(shelf[0].pages));
}

void Read(bookT *favorites, bookT dictionary, int *numBooks)
{
   favorites++;
   favorites[1].pages = -75;
   dictionary.title = favorites[0].title;
   *numBooks -= 50;
   favorites = &dictionary;
         /* Draw diagram with state of memory at this point */
}
```

Problem 4: Data structures (32 points)

You are given the following constant definitions and data structure declarations for use in managing the classes and students of a university:

```
#define MaxUnits 20          /* student can take at most 20 units */
#define MaxStud 5000
#define NumDepts 10

typedef enum {Art, Chem, CS, Econ, EE, English, Math, ME,
              PoliSci, Psych} deptT;

typedef struct {
    string title;     /* title of the class */
    deptT dept;       /* what dept it belongs to (Econ, Math, ..) */
    int nUnits;       /* how many units it is */
    int nEnrolled;    /* how many students are enrolled in class */
    int capacity;     /* max number of students that can enroll */
} classT;


typedef struct {
    classT **schedule; /* dynamic array of pointers to classes */
    int nClasses;      /* number of entries in above array */
    int nUnits;        /* total units for all classes in array */
} studyListT;


typedef struct {
    string name;      /* student name */
    studyListT sl;    /* study list of classes student is taking */
} studentT;


typedef struct {
    classT *class;             /* dynamic array of class records */
    int nClasses;              /* number of entries in above array */
    studentT student[MaxStud]; /* array of student records */
    int nStudents;             /* number of entries in above array */
    classT *best[NumDepts];    /* "best" course by dept */
} universityT;
```

Take a careful look of the data structure definitions, in particular noting where you have pointers and where you have records. You may find it helpful to tear this page out of the exam and use it as reference when answering the three-part question that follows.