Question 1. [6 MARKS]

Part (a) [5 MARKS]

The C program below has errors. When run, the output was:

```
123456789Anwar Patel: Anwar Patel  completed 10 courses with CGPA: 3.700000
123456789Anwar Patel: Anwar Patel  completed 10 courses with CGPA: 3.700000
```

Make changes by printing neatly directly on the code (or beside it) so that it will work as intended. Change as little as necessary. The problem is not the math in the calculation.

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

struct student {
    char id[9];
    char name[20];
    double CGPA;
    int courses;
};

void print(struct student s) {
    printf("%s: %s\t completed %d courses with CGPA: %f\n", s.id, s.name, s.courses, s.CGPA);
}

void complete_course(struct student s, double mark) {
    /* calculate the new CGPA  by weighting the old CGPA appropriately */
    double new_CGPA = ((s.CGPA * s.courses) + mark ) / (float) (s.courses + 1);
    s.CGPA = new_CGPA;
    s.courses++;
}

int main() {
  /* create a record for Anwar Patel (student number 123456789) who has a CGPA of 3.7 on
     his first 10 courses. */

  struct student s1;
  strncpy(s1.id, "123456789", 9);
  strncpy(s1.name, "Anwar Patel", strlen("Anwar Patel"));
  s1.CGPA = 3.7;
  s1.courses = 10;

  print(s1);

  complete_course(s1, 4.0);       /* Anwar just got a 4.0 in CSC209! */

  print(s1);
  return 0;
```

```
}


id needs 10 characters: id char[10]
in strncpy for id need to copy 10
in strncpy for name need strlen("Anwar Patel") + 1
need s1.courses set to 10
function complete_course needs to have parameter struct student *
inside complete_course need to use -> notation or dereference s (now a pointer)
in call to complete_course, need to pass &s
```

Part (b) [1 MARK]

Would your code above still work if the name of the student was "Clement Schiano de Collela""? Explain why or why not and any additional changes to the code that would be required. The above solution will not work correctly because the strncpy will not correctly add a null-terminator to the name. We would need to make sure we copy at most MAXNAME characters in the strncpy statement and then add the

```
s->name[MAXNAME] = '\0';
```

after it.

Question 2. [7 MARKS]

In each subquestion below, fill in the box with the declaration for an appropriate mystery function so that the following code would compile without error. Each subquestion is independent.

Part (a) [3 MARKS]

```
int mystery1(int, char *** );
```

```
char ** students;
int class_size = 100;
if(mystery1(class_size, &students) > 0)
    printf("There were errors\n");
```

Part (b) [4 MARKS]

```
char * mystery2(int, int **, char *);
```

```
char ** authors;
char * most_famous;
int books[4] = {4,12,16,22};
int * p = books;

most_famous = mystery2(books[0], &p, *authors);
```

Question 3. [7 MARKS]

Write a shell program that you might use to clean up your CSC209 directory. The script takes any number of filenames as command line arguments and moves all of the files with names ending in `.sh` to a subdirectory called `scripts`. If this subdirectory doesn't already exist, your script should create it. If your script is called with an argument that doesn't correspond to an existing regular file, you should just ignore that argument. You do not need to worry about further error checking. For example, you may assume that any regular files corresponding to the commandline arguments are readable.

```
#!/bin/sh

# For each of the files in the commandline args, put them in a directory
# based on filename extension of .sh

for file in "$@"; do
  if [ -f "$file" ] ; then
    if [ `expr match "$file" '.*\.sh$'` -gt 3 ] ; then
      if [ ! -d scripts ] ; then
          mkdir scripts
      fi
      mv "$file" scripts/
    fi
  fi
done
```

Question 4. [5 MARKS]

Consider the C program below with an array of strings `names`. Write a C function `zero_strings` that will set each name to an empty string and return the number of names that were changed in the process. Inside the box below, fill in the call to your function inside `main`. You need to determine the appropriate signature for your function.

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

int zero_strings(char ** a, int size) {
    int i;
    int count = 0;
    for (i = 0; i < size; i++ ) {
        if (*a[i] != '\0') {
            count++;
            *a[i] = '\0';
        }
    }
    return count;
}

int main() {
    int n =  XXXXX NOT SHOWN ON TEST XXXX
    char ** names = malloc(n * sizeof(char*));
    int i;
    for (i = 0; i < n; i++ ) {
       names[i] = malloc(NAMESIZE);
    }
    /* code not shown where some of names get set to non-empty strings */

   /* call your zero_strings function and print the return value */

printf("%d\n",zero_strings(names, n));

    return 0;
}
```

Print your name in this box.