CSE 333 2013 Autumn Quiz 1

Date: 10/24/2013

Student ID:_____ Name:_____

1. The following C code is in file q1.c:

```
#include "q1.h"

int main(int argc, char *argv[]) {
  for ( char *p = argv[0]; *p; p++ ) {
    printf("%c", toUpper(*p));
  }
  printf("\n");
  return EXIT_SUCCESS;
}

char toUpper(char c) {
  if ( c >= 'a' && c <= 'z' ) c = c + 'A' - 'a';
  return c;
}
```

Here is what happens when I try to compile and run it:

```
$ gcc -Wall -std=gnu99 q1.c
$ ./a.out
./A.OUT
```

Give plausible contents of the file q1.h.

#include <stdio.h>
#include <stdlib.h>
char toUpper(char c);

2. Suppose a novice C programmer fails to include `#ifdef` guards in the `.h` files he writes. Miraculously, except for that one failing his code is otherwise perfect. Circle all the things that could happen when he tries to build his application:

(1) The code gets a fatal error at preprocessor time and never reaches compile time.
(2) The code preprocesses fine, but gets fatal compiler errors.
(3) It builds fine.

All three are possible.

3. Here is some C code:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  char s[] = "test string";
  for ( char *p = s; *p && *(p+1); p++ ) {
    char c = *p;
    *p = *(p+1);
    *(p+1) = c;
  }
  printf("%s\n", s);
  return 0;
}
```

What happens when I compile and run this code? (Circle one.)

(A) It crashes without printing anything.
(B) It prints "`test string`".
(C) It prints "`est stringt`".
(D) It prints "`ettss rtni`".

(C) happens – it moves the first character to the end of the string, shifting the other characters down.

4. You have an application in C that needs to manipulate many one dimensional arrays of integers. Each such array has a name, represented as a C string. In a full implementation we'd have a method that let us fetch the array's name, as well as one that let us fetch the integervalue of rth element. We won't build those in this question, though.

(Read all three parts before answering.)

(A) Give a C statement that defines a type "NamedArray" that would be suitable for representing this data.
(B) Give code for a method named CreateNamedArray that returns a pointer to a newly created NamedArray. NamedArray takes two arguments: a string name, and an integer number of elements. (You must copy the name string, not simply point at the one the caller passes to you.)
(C) Give code for a method named DestroyNamedArray that takes a pointer to a NamedArray and destroys it.

(A)
```
typedef struct named_array_st {
  char *name;
  int *elements;
} NamedArray;
```

(B)
```
NamedArray * CreateNamedArray(char *name, int nElements) {
  NamedArray *na = (NamedArray*)malloc(sizeof(NamedArray));
  if ( na == NULL ) return NULL;
  na->name = strdup(name);
  if ( na->name == NULL ) {
    free(na);
    return NULL;
  }
  na->elements = (int*)malloc(nElements * sizeof(int));
  if ( na->elements == NULL ) {
    free(na->name);
    free(na);
    return NULL;
  }
  return na;
}
```

(C)
```
void DestroyNamedArray(NamedArray *na) {
  free(na->elements);
  free(na->name);
  free(na);
}
```