# Midterm Exam  Spring 2024



This exam consists of 4 substantive questions. You have 50 minutes – budget your time wisely. Assume the ArmLab/gcc217 environment unless otherwise stated in a problem.

Do all of your work on these pages. You may use the provided blank spaces for scratch space, however this exam is preprocessed by computer, so for your final answers to be scored you must write them inside the designated spaces and fill in selected circles and boxes completely (⚫ and ⬛, not ✔ or ✘). Please make text answers dark and neat.

Name: _____ NetID: _____

Precept:

| | | |
|---|---|---|
| ◯ P01 / P02 - MW<br>Xiaoyan Li | ◯ P10 - TTh 12:30<br>Dwaha Daud | ◯ P07 TTh 2:30<br>Nanqinqin Li |
| ◯ P03 - TTh 12:30<br>Donna Gabai | ◯ P05 - TTh 1:30<br>Donna Gabai | ◯ P08 TTh 3:30<br>Indu Panigrahi |
| ◯ P04 - TTh 12:30<br>Guðni Nathan Gunnarsson | ◯ P06 - TTh 1:30<br>Austin Li | ◯ P09 TTh 7:30<br>Gongqi Huang |

This is a closed-book, closed-note exam, except you are allowed one one-sided study sheet. Please place items that you will not need out of view in your bag or under your working space at this time. Electronic devices such as cell phones, laptops, smartwatches except to check the time, etc. may not be used during this exam.

This examination is administered under the Princeton University Honor Code. Students should sit one seat apart from each other and refrain from talking to other students during the exam. All suspected violations of the Honor Code must be reported to honor@princeton.edu.

In the box below, copy **and** sign the Honor Code pledge before turning in your exam:
*"I pledge my honor that I have not violated the Honor Code during this examination."*

X_____

## Question 0: *Logistics* 0 points

Please don't make the course staff's life harder: make sure you have filled out the fields for your name, your NetID (not PUID, not email alias), precept and the Honor Code pledge text on the front page. Sign your name once you have finished the exam.

## Question 1: *Half As Interesting* as the subsequent questions? 12 points

a. What are the values of the variables in this code after the loop terminates?

```
int x = 0;
int y = 0;
while(x++ < 3)
   y += ++x;
```

x:

y:

b. What is the base 10 value of the variable in this code after the assignment?

```
int z = (6 << 5) | (6 << 2) | -~0;
```

z:

c. In at most 10 words, describe what the `Amystery` function does (i.e., for what property of its parameters does it return `1`):

```
int Amystery(int i1, int i2) { return (i1 ^ i2) < 0; }
```

## Question 2: We *train*ed for this: we *Wendover* `gcc`'s stages a lot 6 points

Consider the following C code:

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

enum {N = 10};

/* print the contents of arr, separated by tabs */
void printarr(int arr[N]) {
   size_t i;
   for(i = 0; i < (size_t) N; i++) {
      printf("%d\t", arr[i]);
   }
}
```

a. Which of the 4 build stages processes the line `#include <stdio.h>`?

| | | | |
|---|---|---|---|
| Preprocessor | ◯ | ◯ | Compiler |
| Assembler | ◯ | ◯ | Linker |

b. Which line is **not** likely added to the code when processing `#include <stdio.h>`?

| | | | |
|---|---|---|---|
| `extern int getchar (void);` | ◯ | ◯ | `#define _STDIO_H 1` |
| `#define EOF (-1)` | ◯ | ◯ | `void printarr(int arr[N]);` |

c. Which is the first stage that would produce a warning or error if the line `#include <stdio.h>` were omitted?

| | | | |
|---|---|---|---|
| Preprocessor | ◯ | ◯ | Compiler |
| Assembler | ◯ | ◯ | Linker |

d. The function **`printarr`** has a major problem that may result in behavior that is not expected by the client. Describe the issue in at most 2 sentences.

## Question 3: ~~*Jet*~~ *Set Lag: The Game* 14 points

Consider the following plausible addition to your String library from Assignment 2:

```
/* Change the char at existing index i of pc to be c */
void Str_set(char *pc, size_t i, char c) {
   assert(pc != NULL);
   if(i < Str_getLength(pc))
      pc[i] = c;
}
```

Further consider some client code using this function – assume these lines appear inside a function body, that all necessary headers have been `#include`d, and that all required memory cleanup occurs later in the function:

```
char ac[] = "Denby";
char *temp;
char *pc1 = "The Boys";
char *pc2 = (char *) malloc(8);

Str_set(ac, 0, 'H');
Str_set(ac, 3, 'd');

Str_set(pc1, 4, 'G');
Str_set(pc1, 5, 'u');

if(pc2 != NULL) {
   Str_copy(pc2, "McManus");
   temp = Str_search(pc2, "M") + 2;
   if(temp != NULL) Str_copy(temp, "Khare");
   Str_set(pc2, 1, '.');
}
```

a. Rewrite the assignment `pc[i] = c;` on the last line of `Str_set`'s body using equivalent pointer notation instead of array indexing bracket notation:

b. The argument 8 to `malloc` is the correct amount to request for this program on armlab. Why is 8 the correct number of bytes to allocate?

- ◯ 8 is the result of `Str_getLength("McManus")` on armlab
- ◯ 8 is the result of `sizeof("McManus")` on armlab
- ◯ 8 is the result of `sizeof(char *)` on armlab

Each row in the table below considers a variable from the client code on page 4. You need **not** answer the grayed out cells. (A bare array name is often treated as a pointer to the 0th array element, but is not actually a pointer variable; `temp` has no `Str_set` call.)

c. In the LOCATION column, indicate in which memory section the variable resides. If the variable's declaration results in a compile-time error, write "ERROR".

d. In the TARGET column, indicate which memory section that variable *references* (i.e., where does it point) after the four declarations at the top of the code.

If there is not enough information to know at that point, or if it is not deterministic, write "NEI". If the variable's declaration does not compile, write "ERROR".

e. In the CONTENTS column, indicate the string's contents through the first `'\0'`, as they are in memory immediately **after** the `Str_set` call(s) for that string.

Assume for the last row in this column that execution does reach the call for `pc2`. If a call results in contents that are nondeterministic, indicate this with "NEI". If a call results in a runtime crash, indicate this with "ERROR".

| | LOCATION | TARGET | CONTENTS |
|---|---|---|---|
| `ac` | | | |
| `temp` | | | |
| `pc1` | | | |
| `pc2` | | | |

## Question 4: Memory Management *Crime Spree* 18 points

Consider the following C code, which you may assume `#include`s all required `.h` files:

```
struct S {
   size_t ulCount;
   int *pi;
};

enum {N = 9};

/* set ps's pi field to a new array {1, 2, ..., 9}
   whose memory will be owned by the caller, and
   set ps's ulCount field to be the array's length */
void S_initSeasons(struct S *ps) {
   size_t ulIndex;
   free(ps->pi);
   (*ps).pi = malloc(N * sizeof(int));
   for(ulIndex = 1; ulIndex <= (size_t) N; ulIndex++)
      (ps->pi)[ulIndex] += ulIndex;
   (*ps).ulCount = N;
}
```

a. The function `S_initSeasons` has *many* problems. In the box below, describe **3** bugs, each in no more than 10 words. Only the first 3 bugs listed will be graded.

   A bug in this problem is something that may:
   - cause a compiler warning or error
   - cause a runtime crash
   - constitute a memory management error
   - violate the specified behavior from the function's comment

Now consider a second function that uses the `struct S` definition from page 6:

```
void S_addSeason(struct S *ps) {
   size_t ulNew;
   assert(ps != NULL);

   ulNew = ps->ulCount + 1;

   /* insert code here */

   ps->pi[ps->ulCount] = ulNew;
   ps->ulCount = ulNew;
}
```

This function updates a structure initialized by a previous call to (a correct version of) **`S_initSeasons`**. Consider these three possible completions for the function at the location designated by the comment in the code:

b. `realloc(ps->pi, ulNew * sizeof(int));`

c. `ps->pi = realloc(ps->pi, ulNew * sizeof(int));`

d.
```
{
   int *pi; /* declaration at the top of an inner block is okay */
   pi = realloc(ps->pi, ulNew * sizeof(int));
   if(pi != NULL) ps->pi = pi;
}
```

In the table below, each row specifies a potential result of the `realloc` call. For each potential result, indicate whether each of the three possible completions would end up with a correct update to `ps` (“OK”) or would end up in a state that would lead to a memory error (“BAD”).

| | b. | | c. | | d. | |
|---|---|---|---|---|---|---|
| | OK | BAD | OK | BAD | OK | BAD |
| Succeeds without relocation | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Succeeds but relocates | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Fails | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |

(Question 4 was the last question – enjoy your *mandatory rest period* … er … spring break! This page is intentionally left blank. Its *Extremities* may be used for scratch work from any problem, however any answers given on this page will not be graded. If you finish the exam early, feel free to use this space to sketch a *nebula* or write a treatise extolling the virtues of *a local snack* from your hometown.)