

## CS 2021: Practice Exam 1

Spring 2021

University of Minnesota

Exam period: 20 minutes

Points available: 50

**Problem 1 (15 pts):** Nearby is a small C program which makes use of arrays, pointers, and function calls. Fill in the tables associated with the approximate memory layout of the running program at each position indicated. Assume the stack grows to **lower memory addresses** and that the sizes of C variable types correspond to common 64-bit systems.

```

1 #include <stdio.h>
2 void flub(double *ap, double *bp){
3     int c = 7;
4     if(*ap < c){
5         *ap = bp[1];
6     }
7     // POSITION B
8     return;
9 }
10 int main(){
11     double x = 4.5;
12     double arr[2] = {3.5, 5.5};
13     double *ptr = arr+1;
14     // POSITION A
15     flub(&x, arr);
16     printf("%.1f\n",x);
17     for(int i=0; i<2; i++){
18         printf("%.1f\n",arr[i]);
19     }
20     return 0;
21 }

```

POSITION A

Frame	Symbol	Address	Value
main()	x	#3064	
	arr[1]	#3056	
	arr[0]		
	ptr		
	i		?

POSITION B

Frame	Symbol	Address	Value
main()	x	#3064	
	arr[1]	#3056	
	arr[0]		
	ptr		
	i		?
flub()			
			7

**Problem 2 (10 pts):**

Fill in the following table of equivalent ways to write these 8 bit quantities. There are a total of 9 blanks to fill in and the first column indicates which blanks occur in which lines. Assume two's complement encoding for the signed decimal column.

	Blank #s	Binary	Hex	Octal	Unsigned Decimal	Signed Decimal
#1 #2 #3		0001 1011		0033		
#4 #5 #6			0xA5	0245		
#7 #8 #9			0xC7			-57

**Background:** Write a short C code fragment (1-5 lines) using a C I/O function call to accomplish the stated task. Assume in each case there is a variable `FILE *fh` which has been opened appropriately for the I/O operation. Also assume that any variables mentioned have already been declared.

**Problem 3 (5 pts):** Read three text floating point values from `fh` formatted as standard decimal point values into double variables `r,u,t`. Check if the read fails due to reaching the end of file; if so print the message `None left`.

**Problem 4 (5 pts):** Write 8 characters from the beginning of array `str` to `fh` in binary format.

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

// Struct to count positive/negative
// numbers in arrays.
typedef struct {
    int poss, negs;
} pn_t;

pn_t *get_pn(int *arr, int len);
// Allocates a pn_t and initializes
// its field to zero. Then scans array
// arr increment poss for every 0 or
// positive value and negs for every
// negative value. Returns the pn_t
// with poss/negs fields set. If arr
// is NULL or len is less than 0,
// returns NULL.

int main(){
    int arr1[5] = {3, 0, -1, 7, -4};
    pn_t *pn1 = get_pn(arr1, 5);
    // pn1: {.poss=3, .negs=2}
    free(pn1);

    int arr2[3] = {-1, -2, -4};
    pn_t *pn2 = get_pn(arr2, 3);
    // pn2: {.poss=0, .negs=3}
    free(pn2);

    int *arr3 = NULL;
    pn_t *pn3 = get_pn(arr3, -1);
    // pn3: NULL

    return 0;
}
```

**Problem 5 (15 pts):** Nearby is a `main()` function demonstrating the use of the function `get_pn()`. Implement this function according to the documentation given. *My solution is about 12 lines plus some closing curly braces.*