Name: JHEID: Score:

Homework 4

CAUTION

- You are expected to work individually.
- **Due: Friday October 9th at 11pm EDT (Baltimore time).**
- *This assignment is worth 20 points.*

SUBMISSION REQUIREMENT

Answer each problem in this **pdf**, in the area to the side of the problem, or immediately after it. You may either type your solutions, or hand-write and scan them in, but they need to be legible, and part of this document. If you need to add additional sheets, please make a note near the problem itself that the grader should "see attached". Submit the **pdf** document via GradeScope once you have added your answers.

Learning Objectives

OBJECTIVES

- pointers, pointers arithmetic
- dynmaic allocation
- type conversions
- bitwise operations
- arrays

INFO

Many problems make use of "code fragments", which can be thought of as pieces of code extracted from complete programs. While a code fragment will not generally compile by itself, we will assume that it exists in a sensible framework (i.e. is inside a properly formed 'main()'', all appropriate headers and libraries have been included, etc.). We will also assume that there is no other code in the program that would impact the behavior of the fragment; each fragment is designed to be understood in isolation.

Part I: Code Writing

Write a code fragment to perform the specified task; remember, it's a fragment, so you don't have to show the include statements or main(), but you should write proper C code that could be compiled and run correctly.

1. You are given an array **arr** of size **n-1** which has unique, unordered, positive integers between **1, 2, ..., n** except one of the numbers is missing. The function should return the missing number by conducting **only one pass of the array 'arr'**. [4 points]

EXAMPLES

Example 1: arr = 1, 3, size = 3 → function returns 2, as the number 2 is missing from the array.

Example 2: arr = 6, 2, 1, 3, 5, size = 6 → function returns 4, as the number 4 is missing.

TIP

You can assume that **arr** has valid positive integers only, **n >= 2** and | **arr** | = **n - 1**.

```
/** Function to find the missing value between {1, ..., n} in arr.
*  @param arr is an interger array of length n - 1 with unique values
*  in {1, ..., n}
*  @param n is the total number of values in the array.
*  @return the missing integer.
*/
int findMissingValue(int *arr, int n) {
```

ANSWER:

ANSWER (continued):

```
}
```

2. Suppose you have a node struct defined as below: [5 points]

```
typedef struct _node {
    Node *next;
    char *str;
} Node;
```

You are given a function that takes in two Node* pointers to two different linked lists. Each linked list node has a string represented by a character array that is properly null terminated. You're function is supposed to return true if the concatenation of strings in a linked list match each other, else return false.

EXAMPLES

List 1: "h" -> "el" -> "l" -> "o" -> NULL
List 2:: "he" -> "llo" -> NULL
Return: True

List 1:: "je" -> "lc -> "l" -> "o" -> NULL
List 2:: "h" -> "e" -> "ll" -> "" -> "o" -> NULL
Return: False

List 1:: "e" -> "llo" -> "h" -> NULL
List 2:: "h" -> "e" -> "l" -> "l" -> "o" -> NULL
Return: False


```
bool areListsMatching(Node *list1, Node *list2) {
```

ANSWER:

```
}
```


Part II: Code Explanation

Give an explanation, in complete English sentences, of the following code snippets. The explanation should be sufficient for a classmate to understand what the code does, even if they had not previously been exposed to it. [1.5 point]

1. [1.5 point]

```
int function1(int num1, int num2) {
    int num = num1 ^ num2;
    int result = 0;
    while(num > 0) {
        result += (num & 1);
        num <<= 1;
    }
    return result;
}
```

ANSWER:

2. [1.5 point]

```
int function2(unsigned int num) {
    return num & ~(num - 1);
}
```

ANSWER:

3. [1.5 point]

```
int f1(unsigned int x) {
  int count = 0;
  while(x) {
      count++; x = x&(x-1);
  }
  return count;
}
```

ANSWER:

Part III: Essay / Short Answer

1. `double ldexp(double x, int exponent)` is a C math function that returns **x** multiplied by **2** raised to the power of **exponent**. How many narrowing and how many promotions happen automatically in the following code line? Name them one by one and explain each one separately. [1.5 points]

```
float f = 2.2f * ldexp(24.4, 3.0 * 2) - 4.4;
```

ANSWER:

2. What is the problem with the following code? Explain. [1 point]

```
#include <stdio.h>
int main() {
    int *p = (int *)malloc(sizeof(int));
    p = NULL;
    free(p);
}
```

ANSWER:

Part IV: Code Tracing

Trace through the code fragment below and write down the exact output that will be printed if the fragment is run. Note that these are called "puzzles" because their behavior may not be intuitive or correct (though the code itself is valid and will compile).

1. [1 point]

```
#include <stdio.h>
int main(void) {
  int a = 10; int *b; int **c = &b; b = &a;
  printf("%d\n", **c++);
  return 0;
}
```

ANSWER:

2. [1.5 points]

```c
#include <stdio.h>
int main() {
  int a[] = {1, 2, 3, 4, 5};
  int *b = a; int **c = &b;
  printf("%d\n", **c*2);
  printf("%d\n", **c-2*4);
  printf("%d\n", **c+1-*b+1);
  return 0;
}
```

ANSWER:

3. [1.5 points]

```c
#include <stdio.h>
int main() {
  int a[][3] = {{1, 2, 3}, {6, 5, 4}, {7, 8, 9}};
  int *b = a[0]; int **c = &b;
  printf("%d\n", b[2]);
  printf("%d\n", **c);
  printf("%d\n", *(*c+3));
  return 0;
}
```

ANSWER:

<The END of Homework 4>

