ENSF 337 – Fall 2022 **Practice Midterm**

Note: In this test you can assume:

- all necessary header files are included, if necessary
- Size of integer data type is 4 bytes, and size of double data type and pointers are 8 bytes, if needed.

SECTION I – FUNCTIONS

Part1 - Consider the following function prototype in C and write the definition/implementation of the function in the following space:

```
char* reverse_strcat(char* dest, char* source);
/* REQUIRES: dest and source each one point to a valid C string.
 * PROMISES: all the characters in source will be appended to the end of dest,
 * in reverse order. Then, it should return the result.
 * EXAMPLES: Assuming dest points to "ABC" and source points to "WXYZ", the function
 * should returns the result which is "ABCZYXW".
 * Notice that "WXYZ" is appended in reverse order.
 */
```

Note: In this function, you can **NOT** use any C library function and you may assume, the space that **dest** points to, is large enough to hold additional characters.

Part 2 - Write a definition for the following C function:

```
bool up_then_down(const int* arr, int n);
// REQUIRES: n >= 1; elements arr[0] ... a[n-1] exist.
// PROMISES: Returns true if the sequence of element values is strictly increasing
// from a[0] to the first appearance of the maximum value, then strictly decreasing
// from the first appearance of the maximum value to a[n - 1].
// Otherwise, returns false. EXAMPLES (maximum values are bold):
// The return value would be true for all of these sequences ...
// {10}, {10, 20}, {20, 10}, {10, 20, 30, 25}, {10, 20, 30, 25, -2}
// But would be false for all of these ...
// {10, 20, 10, 15}, {10, 10, 20, 15}, {10, 20, 20, 15}.
```

Part 3. Write a definition for the following C function. In this part, **you may not make any calls to library functions**.

```
bool all_diff(const char *left, const char *right);
// REQUIRES: left and right each point to the beginnings of C strings.
// PROMISES: Return value is true if none of the characters in the left string
// appear in the right string. ('\0' characters are not included in the
// comparison.)
// If there is at least one match, return value is false.
```

Part 4 – In the following space, write the definition of the function called `compareMyStrings` with the following function prototype:

```
int compareMyStrings (const char* arg1, const char* arg2);
```

This function is supposed to return:

- Zero, if two string arguments of the function are identical
- One, if first string is greater than second string
- Minus one, if second string is greater that first string.

Some examples: if the first argument is "aB" and the second argument is "ABCD" the function should return 1 because the first argument is lexicographically (means dictionary order) greater than second argument. However, if first argument is "AB" and the second argument is "ABCD" returns -1 because the second argument is greater than first argument.

Part 5 - Consider the definition of the C structure called Clock:

```
typedef struct Clock{
    int hours;
    int minutes;
    double seconds;
}Clock;
```

In the following space write the definition of a C function called `millisecond_to_Clock`. This function should have one argument of type `long int`, which represents time in `milliseconds`. Then, it converts the `milliseconds` to an instance of Clock object and returns a pointer to this object. For Example, if `milliseconds` is `18123400`, the converted object of of type `Clock` must have values of: `5` for `hours`, `2` for `minutes` and `3.4` for the `seconds`.

Here is the prototype and function's interface comment:

```
Clock millisecond_to_Clock (long milliseconds);
/* REQUIRES: milliseconds >= 0
 PROMISES: creates an instance of Clock, then assigns the number of hours, minutes and
seconds in the milliseconds to the data members in the local Clock object, and returns its
address.
*/
```

Part 6: The Fibonacci Sequence is the series of numbers, that its first two numbers are 0 and 1, and the folloiwng numbers are:

$$F_n = F_{n-1} + F_{n-2}.$$

Here is the first 10 fibonacci numbers: **0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...**

What to Do:

Consider the following interface comment for function **is_fibonacci_sequence** and write the COMPLETE implementation of the function in the following space.

```
int is_fibonacci_sequence(const int* x, int n);
/* REQUIRES: x points to an integer array with n elements.
 * PROMISIS:
 * - if n is less than 2 returns zero.
 * - if first element of array is not 0 and second element is not 1
 *   returns zero.
 * - Otherwise, returns 1 if elements x[2] to x[n-1] are
 *   equal to sum of the previous elements.
*/
```

SECTION II – ARRAYS, POINTERS

Draw an AR diagram for point ONE in the following program.

```
long int function(char s[2], char p[1]){
    long int y = p - s;
    // POINT ONE
    return y;
}

int main() {
    int i = 0;
    const char* str1 = "ABC";
    char str2[] = "XY";
    char str3[4] ;
    str2[0] = *(str2 + 1);
    while(*str1){
        str3[i] = *str1 + 1;
        str1++;
        i++;
    }

   function(str3, str3 + i);
   return 0;
}
```

SECTION III

Assuming all required header file are included, draw a memory diagram for the following Program at point 1.

```
struct Type {
    const char* bar;
    int num;
} ;

char f(Type arg1, Type *arg2){
    const char *p;
    arg2->num = 663;
    p = &arg2->bar[2] - 1;
    char c = *arg1.bar ;
    /* point one */
    return c;
}

int main(void){
    Type foo = { "FM", 8 };
    Type* Q = new Type;
    *Q = foo;
    char c = f(foo, Q);
    c++;
    return 0;
}
```

Section IV – Short Answer Questions

Part a. What is the output from the following code fragment?

```
int a[5] = { 2 - 2, 1, 0, 1, -2 }, i;
  for (i = 0; i < 5; i++) {
    if (a[i])
     printf("Y");
   else
    printf("N");
}
```

Answer:

Part b. An executable file is made from the two rather bizarre files below. What is the output of the program?

Answer:

File main.c

```
#include <stdio.h>
int main(void)
{
#include "stuff.h"
#include "stuff.h"
  return 0;
}
```

File stuff.h

```
printf("hello!\n");
#ifndef GOODBYE
#define GOODBYE "bye!"
printf("GOODBYE %s\n", GOODBYE);
#endif
```

Part c. What is the output of the program if the size of an int is 4 bytes and the size of a pointer is 8 bytes?
Answer is:

What is the output of the program if the sizes of ints and pointers are both 4 bytes
Answer is:

```
#include <stdio.h>
void func(int y[5]);
int main(void) {
   int x[5] = {10, 8, 6, 4, 2};
   printf("main says %lu\n",
   sizeof(x)/sizeof(x[0]));
   func(x);
   return 0;
}
void func(int y[5]) {
   printf("func says %lu\n",
sizeof(y)/sizeof(y[0]));
}
```

Part d. What is the output from the following code fragment? For full credit, you must show how you got your answer.

```
char x[10] = {
 'A', 'B', 'C', '\0', '\0',
 '\0', '\0', '\0', '\0', '\0'
};
strcat(x, "UV");
x[8] = 'W';
printf("%lu %lu %lu %lu\n",
strlen(x), strlen(x+2), strlen(x+5), strlen(x+8));
```

Answer:

Part e. Assume that the call to malloc succeeds, draw a memory diagram for point one.

```
#include <stdio.h>
#include <stdlib.h>
int glob[3] = { 31, 57 };
int main(void) {
        int loc[3];
        int *dyn = malloc(3 * sizeof(int));
        loc[0] = glob[0];
        dyn[1] = glob[1];

        // point one

        return 0;
}
```

Part f. What is the output of the following program assuming that text file numbers.txt is located in the same working directory and contains the values in the following box:

```
126  456
333  abc  900
```

```
#include <stdio.h>
int main() {
  FILE* fp;
  int a[5] = {-1, -2, -33, 4, 500};
  char filename[20] = "numbers.txt";
  fp = fopen(filename, "r");
  if(fp == NULL){
     printf("File not found.");
     exit(1);
  }
  a[4] = fscanf(fp, "%d%d%d%d", &a[0], &a[1], &a[2], &a[3]);
  printf("%d  %d  %d  %d  %d", a[0], a[1], a[2], a[3], a[4]);
  return 0;
}
```

Write your answer in the following space:

SECTION V – MULTIPLE-CHOICE

1. What is the output of the following code segment?

```
char* s1 = "apple";
char s2[]= "Victor";
int* p = new int [5];
cout << (int)sizeof(s1)  << " " << (int)sizeof(s2) << " "
     << (int) strlen(s2) << " " << (int)sizeof(p)  << endl;
```

a. 5 6 5 8
b. 8 7 6 8
c. 8 8 4 8
d. 6 6 5 8
e. 6 6 6 4
f. None of the above.

2. What if anything is wrong with the following code segment?

```
char code[6] = "Apple";
const char* csp = code + 1;
csp++;
*code = csp[0];
*csp = *code;
```

a. There is a runtime error in this code segment.
b. There is a compilation error on the second-last line of the code segment.
c. There is a compilation error on the last line of the code segment.
d. All of the above are correct
e. None of the above is correct.

3. What is the output of the following code segment?

```
char course[] = "ENCM339F20016";
char* sp = course + 2 ;
while(*sp != '6'){
     cout << *sp;
     sp = sp + 2 ;
 }
```

a. ENM339F201
b. CM33oF201
c. C3920
d. Ec3921
e. None of the above.

4. What is the value of y after this code segment is implemented?

```
long int y;
const char* s= "012345678";
const char* sp = &s[10];
y = s - sp;
```

a. Garbage
b. 10
c. -10
d. 9
e. -9

5. What is the output of the following code segment?

```
char s1[10] = "ENCM";
char s2[10] = "PHYS";
  printf("%s\n", strcpy(s1, strcat(s1, s2)));
```

a. ENCMPHYS
b. PHYS
c. ENCM
d. PHYSENCM

6. Consider the following code segment?

```
int numbers [7] = {11, 22, 33, 44, 55, 66, 17};
int *p = &numbers[5];
printf("%d\n", (*p -*numbers));
```

Which one of the following is the correct output?

a. 6
b. 5
c. 66
d. 55
e. 22

7. What is the output of the following code segment?

```
char * st1 = "ENCM 339";
int i = 0;
st1 = st1 + 2;
printf("%c\n", st1[--i]);
```

a. N
b. M
c. NULL
d. C

8. Consider the following code segment:

```
int i, j;
printf ("%d\n", scanf("%d%d", &i, &j));
```

What is output of the program if the user's input (on the keyboard) is: `56 44`

a. 56
b. 44
c. 2
d. 56 44

9. Consider following structures and the main function.

```
struct Info {
        char fname[20];
        char lname[30];
        int age;
};
struct Player {
       Info p_infor;
       double shot_accuracy;
       double speed;
};

int main() {
    Player flames[20];
    Player *p;
    p = flames;
    return 0;
  }
```

With regard to above code, which one of the following lines of code can be used to change "age" for the fist index of the array flames?

a. `(*p).p_info.age = 32;`
b. `(*p)->p_info.age = 32;`
c. `p.p_info.age = 32;`
d. `flames->p_info.age = 32;`
e. `p.p_info->age = 32;`

10. Consider the following small program. Which one of the calls to the function foo is correct

```
void foo (int *x, int *y, int *z){
     *z = (*x) * (*y);
}

int main (){
     int a [5] = {10, 20, 30};
     int *b = a;
     int c = 5;

     //Function call goes here
}
```

a. `foo (a[0], b[1], &c);`
b. `foo (a, b, &c);`
c. `foo (a[], b[], &c);`

d. foo (&a, &b, &c);

```
int main(){
    char *s = malloc(30);
    strcpy(s, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXX"); // 29 Xs
    float* p = (float*)s;
    p+=2;
    // point one
    return 0;
}
```

11. Consider the following main function. what is the value of p at point one.

a) &s[4]
b) s[8]
c) &s[8]
d) s[0]
e) &s[0]
f) s
g) s[4]