CSE 220 – C Programming

Exam Review

Time and Organization

- Wednesday 15:00 PM – 15:50 PM
- Mimir homework
  - Stable Internet connection, no extension allowed.
  - Submit your answers before leave.
  - Remind you at 15:49 PM, automatically submit at the end
- Zoom Link
  - Session 1: https://msu.zoom.us/j/6269778746 (Passcode: EB3224)
  - Session 2: https://msu.zoom.us/j/5462365219 (Password: cse220-lab)
- Breakout rooms
  - Share your screen and open your camera.
  - Join at least 2 minutes earlier.

Can and Can't

- Do it by yourself
- Can
  - Lecture slides
  - Lab codes
  - Mimir IDE or other online C programming environment
  - Online materials
- Can't
  - Put your answer to online discussion group or forum
  - Cooperation through mobile Apps

Grading

- 7 problems, each 3 points
- Long answer format as you experienced in homework
  - Manually grading
- The highest 6 points will be counted
- Earn at least 9 points

Exam Scope

- All contents discussed before (https://www.cse.msu.edu/~cse220/schedule-20-fall.html)

Date (Week)	Monday Lecture	Wednesday Lecture
2020-08-31	No Lecture (Classes Haven't Started)	Introduction
2020-09-07	No Lecture (Labor Day)	C Fundamentals
2020-09-14	C Fundamentals Part 2	Input/Output
2020-09-21	Expressions	Conditional Statements
2020-09-28	Loops	Loops continued
2020-10-05	Data types	Arrays
2020-10-12	Functions	Functions continued
2020-10-19	Pointers	Pointers continued (Middle of Semester)
2020-10-26	Pointers and Arrays	Pointers and Arrays continued
2020-11-02	Strings	Command Line Arguments
2020-11-09	Exam Review	Exam

Problem Types

- Legal or Illegal?
  - Statements, Expressions, Declaration and Initialization
- Coding
  - Function
  - Pointer
  - Array
  - String
- Output
  - Data types
  - Recursive Function
  - Pointer and Array

Legal or Illegal

Legality (Exam Sample 1)

(a) (1 point) Which of the following contain valid strings?

- √ `char * x = "abc";`
- √ `char y[] = "a'c'd";`
- ◯ `char x7 = "a\n5.?";`
- ◯ `char * dog = 'ab';`
- √ `char * ptr = {'a', 'j', '\n', '\0'};`
- √ `char example[3] = {'4', '\0'};`
- √ `char z[5]; z[1] = 'y'; z[2] = '\0'; z[0] = '5';`
- √ `char cat[5]; cat[0] = '\0'; char *ferret = cat;`

Full credit for 2 or fewer wrong, half credit for 4 or fewer wrong.

(b) (1 point) Which of the following are legal C statements?

- √ `int apple = 'c';`
- ◯ `char josh[3]; josh[3] = 'c';`
- √ `float dog; float * f = &dog;`
- ◯ `double wish[2] = {4.5, 7.7, 11.3}`

Coding

No Indexing (Exam Sample 1)

You need to write a function that counts the number of spaces, and non-space characters in a string. The function delivers these counts through two pointers to int passed in as arguments. Here is its function declaration:

```
void count(char * str, int * spaces, int * non_spaces);
```

Example use:

```
int main(void) {
    char * string = "my\nname is Josh.";
    int a = 11, b = 6;
    count(string, &a, &b);
    // a should now be 2, b should now be 14
    return 0;
}
```

You need to write the function (named "count"), but **you are not allowed to use the characters `[` or `]`.**

No Indexing (con't)

Solution:

```
void count(char * str, int * spaces, int * non_spaces) {
    *spaces = 0;
    *non_spaces = 0;
    for (char * ptr = str; *ptr != '\0'; ++ptr) {
        if (*ptr == ' ') {
            ++(*spaces);
        } else {
            ++(*non_spaces);
        }
    }
}
```

Repeat String (Exam Sample 2)

Write a function named "repeat", that takes a char * (string) as its only argument. This function should repeat the contents of the string, making the number of non-null characters double. For example, if the input string is "cat", after calling repeat, the string should be "catcat". You may not make any function calls in this function. You may assume the char array holding the string has enough room for the resulting string.

Repeat String (con't)

Solution:

```
void repeat(char * str) {
    char * old_end = str;
    while (*old_end != '\0') {
     ++old_end;
    }
    char * ptr = old_end;
    while(str != old_end) {
        *ptr = *str;
        ++ptr;
        ++str;
    }
    *ptr = '\0';
}
```

Input and Output

Recursion (Exam Sample 1)

Below is a program which uses a recursive function (named "abc"). For each of the supplied inputs, write what the program would output. If the program would perform an illegal action, write "illegal" instead of the output.

```
void abc(int x, char c, char * ptr);
int main(void) {
    char string[5]; int num; char ch;
    scanf("%d %c", &num, &ch);
    abc(num, ch, string);
    printf("%s", string);
}
void abc(int x, char c, char * ptr) {
    if (x == 0) {
        *ptr = '\0';
    } else {
        *ptr = c;
        abc(x - 1, c + 1, ptr + 1);
    }
}
```

Recursion (con't)

```
void abc(int x, char c, char * ptr);
int main(void) {
    char string[5]; int num; char ch;
    scanf("%d %c", &num, &ch);
    abc(num, ch, string);
    printf("%s", string);
}
void abc(int x, char c, char * ptr) {
    if (x == 0) {
        *ptr = '\0';
    } else {
        *ptr = c;
        abc(x - 1, c + 1, ptr + 1);
    }
}
```

(a) (1 point) `1 a` a

(b) (1 point) `4 d` defg

(c) (1 point) `3 4` 456

(d) (1 point) `7 j` **illegal**

Pointers and Arrays (Exam Sample 2)

For each code section, write what would be outputted by the code. If the code performs illegal actions, write "illegal".

(a) (10 points)
```
int j = 'c'; int k = 3;
char str[] = {'A', 'B', 'C', 'D', 'E'};
str[k] = j;
printf("%s", str);
```

illegal (str is doesn't have a null character)

(b) (10 points)
```
int j = 10; int k = 15;
int * m = &j; k = j; *m = 4;
printf("%d:%d:%d", j, k, *m);
```

4:10:4

Pointers and Arrays (con't)

(c) (10 points)
```
char a[] = "1234";
char *b = a; b += 2; *b = '9';
printf("%s %s", a, b);
```

1294 94

(d) (10 points)
```
char a[] = "the end"; char *b = a; int i = 0;
for ( ; b[i] != ' '; ++i) {
    printf("%d--", i);
}
printf("(%d)", (&b[i]) - a);
```

0--1--2--(3)

(e) (10 points)
```
char a[] = "abcd"; char * b = "xyz";
for (int i = 0; b[i] != '\0'; ++i) {
    *(a + i) = b[i] - 1;
}
printf("%s %s", a, b);
```

wxyd xyz