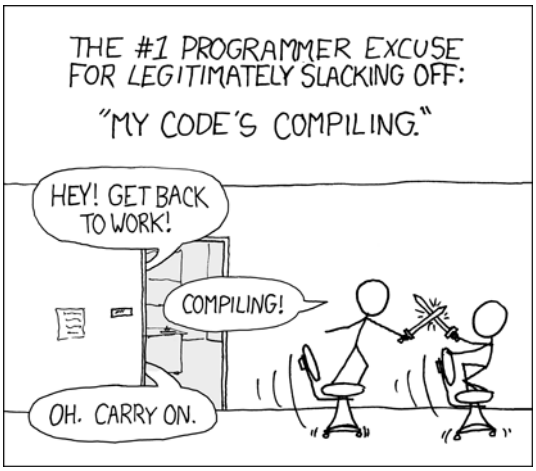
First Name: _____

Autumn 2016 Last Name: _____

# Primary Exam for CSE 220 (2016)

Answer the questions in the spaces provided on the page. If you run out of room for an answer, continue on the back of the page.

- DO NOT OPEN THE EXAM UNTIL TOLD TO DO SO
- You only need to answer the first question and 4 of the 5 remaining questions.
- On one of the questions, make a large slash across the page, which indicates that it should not be graded.
- On every page (including the first and last page), write your first and last name, before answering the question. Unnamed pages may be lost.
- If you start to answer a question and then change your mind, please cross out the attempt and write *DO NOT GRADE* across it.
- Legibility matters! If we can't read your answer, you will receive a 0 for it.




https://xkcd.com/303/

First Name: _____

Autumn 2016 Last Name: _____

Question 1: Legality (REQUIRED QUESTION) ..... *2 points*

Fill in the bubbles of the correct choices.

(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}`

Full credit for 1 or fewer wrong, half credit for 2 or fewer wrong.

Points earned: _____ out of a possible 2 points

First Name: _____

Autumn 2016 Last Name: _____

Question 2: No Indexing ..... *4 points*

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 `]`.**

**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);
        }
    }
}
```

Points earned: _____ out of a possible 4 points

First Name: _____

Autumn 2016 Last Name: _____

Grade Breakdown

- 0.5 pts - correct header
- 0.5 pts - correct pointer usage demonstrated
- 0.5 pts - uses loop
- 0.5 pts - correct range on loop execution
- 0.5 pts - conditional or other separation for spaces/non-spaces
- 0.5 pts - correct value assigned for non-spaces
- 0.5 pts - correct value assigned for spaces
- 0.5 pts - formatting/syntax/other

Points earned: _____ out of a possible 0 points

First Name: _____

Autumn 2016 Last Name: _____

Question 3: Recursion ..... *4 points*

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);
    }
}
```

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

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

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

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

Points earned: _____ out of a possible 4 points

First Name: _____

Autumn 2016 Last Name: _____

Question 4: Pointers and Arrays ..... *4 points*

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

(a) (1 point)
```
int x = 3, y = 1;
int array[] = {2, 5, 6, 11, 13};
int * ptr = &array[x];
ptr += y;
printf("%d", *ptr);
```

13

(b) (1 point)
```
char array_2[4] = {'7', '8'};
array_2[3] = 'a';
printf("%s", array_2);
```

78

(c) (1 point)
```
float array_3[] = {3.0, 5.6, 4.5};
float * p_f = array_3;
++p_f;
array_3 = p_f;
printf("%.1f", array_3[0]);
```

illegal

(d) (1 point)
```
char array_4[10] = "abcd";
char * ptr_2 = array_4;
while (*ptr_2 != 'c') {
++ptr_2;
}
ptr_2 = 'X';
printf("%s", array_4);
```

abcd

Points earned: _____ out of a possible 4 points

First Name: _____

Autumn 2016 Last Name: _____

Question 5: Functions ..... *4 points*

For each of the supplied inputs, write what the program would output.

```
#include <stdio.h>
char dog(char x, int y) {
    char z = x + y;
    if (z > 'z') {
        z -= 26;
    }
    return z;
}
int cat(int * x) {
    return *x;
}
int main(void) {
    int c; char d;
    scanf("%d %c", &c, &d);
    int b = cat(&c);
    char a = dog(d, b);
    printf("%c", a);
    return 0;
}
```

(a) (1 point) `1 e` _____f_____

(b) (1 point) `3 x` _____a_____

(c) (1 point) `3 w` _____z_____

(d) (1 point) `0 j` _____j_____

Points earned: _____ out of a possible 4 points

First Name: _____

Autumn 2016 Last Name: _____

Question 6: Indexing ..... *4 points*

You need to write a function that calculates the sum of the elements in a array, specified by an other array. The first array (named "values") contains integers. The second array (named "indices") is an array of the indices of the values array that should be tallied. The last argument (named "size_of_indices") is the size of the array "indices".

Here is its function definition (incomplete):

```
int sum_indices(int values[], int indices[], int size_of_indices) {
// YOUR CODE HERE
}
```

Example use:

```
int main(void) {
    int values[] = {1, 3, 5, 7};
    int indices[2] = {0, 2};
    printf("%d", sum_indices(values, indices, 2));
    // Should print 6 because:
    // index 0 is 1
    // index 2 is 5
    // the sum of 1 and 5 is 6
    return 0;
}
```

**Solution:**

```
int sum = 0;
for (int i = 0; i < size_of_indices; ++i) {
    sum += values[indices[i]];
}
return sum;
```

Points earned: _____ out of a possible 4 points

First Name: _____

Autumn 2016 Last Name: _____

Grade Breakdown:

- 0.5 pts - initializes sum variable correctly
- 0.5 pts - uses loop
- 0.5 pts - correct range on loop execution
- 0.5 pts - lookup into values
- 0.5 pts - lookup into indicies
- 0.5 pts - correct value for sum
- 0.5 pts - return exists, passes sum
- 0.5 pts - formatting/syntax/other

Points earned: _____ out of a possible 0 points

First Name: _____

Autumn 2016 Last Name: _____

If you have finished early, feel free to bring your exam to an instructor.
Or, you can draw a picture of your favorite Pokémon.
Or, you can write a haiku about your love of multidimensional arrays.

| Question | Points | Score |
|---|---|---|
| Legality (REQUIRED QUESTION) | 2 | |
| No Indexing | 4 | |
| Recursion | 4 | |
| Pointers and Arrays | 4 | |
| Functions | 4 | |
| Indexing | 4 | |
| Total: | 18 | |