Started on	Wednesday, 4 September 2019, 9:21 PM
State	Finished
Completed on	Wednesday, 4 September 2019, 10:24 PM
Time taken	1 hour 3 mins
Marks	13.00/14.00
Grade	**18.57** out of 20.00 (**93**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Write a C program that prints the one-line message *OK so far!* Note that the exclamation mark *is* part of the message to be printed.

For example:

Result
`OK so far!`

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

int main(void)
{
    printf("OK so far!");
    return EXIT_SUCCESS;
}
```

	Expected	Got	
✔	`OK so far!`	`OK so far!`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Recall that a quadratic equation $ax^2 + bx + c = 0$ has complex (or imaginary) roots if b^2 is less than $4ac$ or real roots otherwise. Write a C program that reads three integers *a*, *b* and *c,* representing the coefficients of a quadratic equation, from standard input and prints either the line

```
Roots are real
```

or the line

```
Roots are complex
```

whichever is appropriate. The program then exits. You may assume that the three integers are on a single line of input separated by a single white space character.

For example:

Input	Result
1 2 3	Roots are complex
1 3 2	Roots are real

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

int main(void)
{
    int one;
    int two;
    int three;
    scanf("%d %d %d", &one, &two, &three);
    if (((two*two) - (4 * one * three)) < 0) {
        printf("Roots are complex");
    } else {
        printf("Roots are real");
    }
    return EXIT_SUCCESS;
}
```

	Input	Expected	Got	
✔	1 2 3	Roots are complex	Roots are complex	✔
✔	1 3 2	Roots are real	Roots are real	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Write a program that reads a single integer *n* from standard input and then prints to standard output the cubes of the integers from 1 up to *n* (inclusive), one per line. *n* will always be greater than or equal to 1.

For example:

Input	Result
4	1 8 27 64

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

int main(void)
{
    int one;
    scanf("%d", &one);
    for (int aii = 1; aii <= one; aii++) {
        printf("%d\n",(aii * aii * aii));
    }
    return EXIT_SUCCESS;
}
```

	Input	Expected	Got	
✔	4	1 8 27 64	1 8 27 64	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Write a program that reads in a series of integers until a value of zero is encountered and then prints the sum of all the numbers except those that are equal to 13 or that come immediately after a 13. It is guaranteed that there is at least one zero in the input.

Input: at most 100 lines of input, each line containing a single integer. At least one of the lines will contain the integer 0.

Output: the sum of all the input numbers up to the first zero excluding any numbers equal to 13 or immediately following a value of 13.

Note: the sum of an empty set is zero.

For example:

Input	Result
3 13 10 20 13 9 400 0 7	423

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

int main(void)
{
    int counter = 0;
    int one;
    scanf("%d", &one);
    while (one != 0) {
        if (one == 13) {
            scanf("%d", &one);
        } else {
            counter += one;
        }
        scanf("%d", &one);
    }
    printf("%d", counter);
    return EXIT_SUCCESS;
}
```

	Input	Expected	Got	
✔	3 13 10 20 13 9 400 0 7	423	423	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Write a function with signature

```
int add_even_sub_odd(const int data[], int n)
```

that returns the sum of the even values in the `n`-element array `data` minus the sum of the odd values. In the test case below the result is calculated as:

0 - 1 + 2 - 3 + 4 = 2

since 2 and 4 are even numbers (which get added), and 1 and 3 are odd numbers (which get subtracted).

For example:

Test	Result
`int data[4] = {1, 2, 3, 4};` `printf("%d\n", add_even_sub_odd(data, 4));`	2

Answer: (penalty regime: 0, 10, 20, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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



int add_even_sub_odd(const int data[], int n)
{
    int counter = 0;
    for (int aii = 0; aii < n; aii++) {
        if (data[aii] % 2 != 0) {
            counter -= data[aii];
        } else {
            counter += data[aii];
        }
    }
    return counter;
}
```

	Test	Expected	Got	
✔	`int data[4] = {1, 2, 3, 4};` `printf("%d\n", add_even_sub_odd(data, 4));`	2	2	✔
✔	`int data[4] = {-1, 2, -3, -4};` `printf("%d\n", add_even_sub_odd(data, 3));`	6	6	✔
✔	`int data[6] = {5, -9, -3, 0, 8, -1};` `printf("%d\n", add_even_sub_odd(data, 6));`	16	16	✔
✔	`int data[3] = {-9999, -10001};` `printf("%d\n", add_even_sub_odd(data, 2));`	20000	20000	✔
✔	`int data[2] = {-1, 13};` `printf("%d\n", add_even_sub_odd(data, 0));`	0	0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **6**

Correct

Mark 1.00 out of 1.00

Write a function with signature

```
void swap3(int* x, int* y, int* z)
```

that swaps the values of the integers pointed to by the parameters into a specific ordering. If `x` points to an integer `a`, `y` points to integer `b`, and `z` points to integer `c`, then the integer values should be shuffled such that

$$c \leq b \leq a$$

holds true. You can assume that the three parameters have different values, i.e., they point to different `int` variables.

For example:

Test	Result

`int a = 10;`
`int b = 0;`
`int c = 7;`
`swap3(&a, &b, &c);`
`printf("%d <= %d <= %d\n", c, b, a);`
	`0 <= 7 <= 10`

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

void swap3(int* x, int* y, int* z)
{
    int a = *x; //biggest
    int b = *y; //middle
    int c = *z; //smallest
    if(a > b && a > c) {
        //a is biggest
        if(b > c) {
            //order is c, b, a
            *x = a;
            *y = b;
            *z = c;
        } else {
            //order is b c a
            *x = a;
            *y = c;
            *z = b;
        }
    } else if(b > a && b > c) {
        //b is biggest
        if(a > c) {
            //order is c a b
            *x = b;
            *y = a;
            *z = c;
        } else {
            //order is a c b
            *x = b;
            *y = c;
            *z = a;
        }
    } else if (c > a && c > b) {
        //c is the biggest
        if (a > b) {
            //order is b a c
            *x = c;
            *y = a;
            *z = b;
        } else {
            //order is a b c
            *x = c;
            *y = b;
            *z = a;
        }
    }
}
```

	Test	Expected	Got	

✔	`int a = 10;`
	`int b = 0;`
	`int c = 7;`
	`swap3(&a, &b, &c);`
	`printf("%d <= %d <= %d\n",`
	`c, b, a);`
		`0 <= 7 <= 10`	`0 <= 7 <= 10`	✔

	Test	Expected	Got	

✔	`int a = -2147483648;`
	`int b = 2147483647;`
	`int c = -2147483647;`
	`swap3(&a, &b, &c);`
	`printf("%d <= %d <= %d\n", c, b, a);`
		`-2147483648 <= -2147483647 <= 2147483647`	`-2147483648 <= -2147483647 <= 2147483647`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **7**

Correct

Mark 1.00 out of 1.00

Write a function with signature

```
char* rotateLeft(char* s)
```

that performs a left rotation on the characters in a string *in place* and returns a pointer to the modified string (which will be equal to the pointer to the original string). A left rotation moves each item to its next lower position in a sequence, with the first item being moved to the last position.

For example:

Test	Result
`char s[] = "Hello World!";` `printf("%s\n", rotateLeft(s));`	`ello World!H`

Answer: (penalty regime: 0, 10, 20, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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



char* rotateLeft(char* s)
{
    char lengthofstring = strlen(s);
    char firstletter = *s;
    for(int i=0; (i < lengthofstring); i++) {
        s[i] = s[i + 1];
    }
    s[lengthofstring - 1] = firstletter;
    return s;
}
```

	Test	Expected	Got	
✔	`char s[] = "Hello World!";` `printf("%s\n", rotateLeft(s));`	`ello World!H`	`ello World!H`	✔
✔	`char s[] = "ENCE260";` `printf("%s\n", rotateLeft(s));`	`NCE260E`	`NCE260E`	✔
✔	`char s[] = "C is not so bad after all";` `printf("%s\n", rotateLeft(s));`	` is not so bad after allC`	` is not so bad after allC`	✔
✔	`char s[] = "0123456789";` `printf("%s\n", rotateLeft(s));`	`1234567890`	`1234567890`	✔

✔	`char s[] = "';lkjhgfdsa";`
	`printf("%s\n", rotateLeft(s));`
	`if (s != rotateLeft(s)) {`
	`  printf("Test feedback: Your function's return value is incorrect!\n");`
	`}`
		`;lkjhgfdsa'`	`;lkjhgfdsa'`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **8**

Correct

Mark 1.00 out of 1.00

Write **your own** version of the standard library *strchr* function, with the signature `char* mystrchr(char* s, int c)`. The function should return a pointer to the first occurrence of the character c in the string s or NULL if no matching character exists. Your function must not call any other functions, including of course the library *strchr* function! It also must not use any #define, #include or other preprocessor commands.

Make sure you name your function *mystrchr* not *strchr*.

NOTE: the test case(s) below print the integer obtained by subtracting the string start address from the function return value (unless the latter is NULL). The integer obtained in this way is the subscript into the string. It is printed using a "%zd" format in order to ensure it behaves the same on both 32-bit architectures and 64-bit architectures. Don't let it worry you :-)

For example:

Test	Result

`char* s = "ENCE260";`
`char* foundAt = mystrchr(s, '2');`
`if (foundAt == NULL) {`
`  puts("Not found");`
`}`
`else {`
`  printf("%zd\n", foundAt - s);`
`}`
	4

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
char* mystrchr(char* s, int c)
{
    while (*s != '\0') {
        if (*s == c) {
            return s;
        }
        s++;
    }
    return NULL;
}
```

	Test	Expected	Got	

✔	`char* s = "ENCE260";`
	`char* foundAt = mystrchr(s, '2');`
	`if (foundAt == NULL) {`
	`  puts("Not found");`
	`}`
	`else {`
	`  printf("%zd\n", foundAt - s);`
	`}`
		4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **9**

Correct

Mark 1.00 out of 1.00

Write a function `int* findPair(int* data, int numEls, int first, int second)` that searches within the first `numEls` of an `int` array `data` for an element equal to `first` followed immediately by an element `second`. If this situation occurs, the function returns a pointer to the element `first`. If the situation occurs more than once in the array, the pointer should be to the first occurrence. If the situation does not occur the function should return `NULL`.

[The tests print the result of subtracting two pointers using the "%zd" format so that the code works on both CodeBox and the quiz server. Don't worry about it.]

For example:

Test	Result

`int data[] = {1, 10, 3, 20, 1, 3, 7};`
`int* p = findPair(data, 7, 1, 3);`
`if (p != NULL) {`
`  printf("Found at position %zd\n", p - data);`
`}`
`else {`
`  puts("Not found");`
`}`
	`Found at position 4`

`int data[] = {1, 10, 2, 1, 2, 1, 2};`
`int* p = findPair(data, 7, 1, 2);`
`if (p != NULL) {`
`  printf("Found at position %zd\n", p - data);`
`}`
`else {`
`  puts("Not found");`
`}`
	`Found at position 3`

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
int* findPair(int* data, int numEls, int first, int second)
{
    for (int i = 0; i != (numEls - 1); i++) {

        if (*data == first) {
            if (*(data + 1) == (second)) {
                return data;
            }
        }
        data++;
    }
    return 0;


}
```

	Test	Expected	Got	

✔	`int data[] = {1, 10, 3, 20, 1, 3, 7};`
	`int* p = findPair(data, 7, 1, 3);`
	`if (p != NULL) {`
	`  printf("Found at position %zd\n", p - data);`
	`}`
	`else {`
	`  puts("Not found");`
	`}`
		`Found at position 4`	`Found at position 4`	✔

	Test	Expected	Got	

✔	`int data[] = {1, 2, 1, 1, 1, 1};`
	`int* p = findPair(data, 6, 1, 1);`
	`if (p != NULL) {`
	`  printf("Found at position %zd\n", p - data);`
	`}`
	`else {`
	`  puts("Not found");`
	`}`
		`Found at position 2`	`Found at position 2`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **10**

Correct

Mark 1.00 out of 1.00

Write the code to be inserted at the point marked "*** YOUR CODE GOES HERE ***" such that the program shown below compiles correctly and prints the following output when run:

```
A 30 mL drink of Tequila is about 1.4 standard drinks.
```

IMPORTANT: paste into the answer box *only* the code that replaces the comment '// *** YOUR CODE GOES HERE ***' *not* the whole program.

```
#include <stdio.h>

// *** YOUR CODE GOES HERE ***

int main(void)
{
    Drink shot = {"Tequila", 0.6, 30};
    printf("A %d mL drink of %s is about %.1f standard drinks.\n", shot.volume, shot.beverage,
shot.volume * shot.abv * 0.0789);
}
```

Note: There is no precheck button for this question.

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```

typedef struct {
    char* beverage;
    float abv;
    int volume;
} Drink;
```

	Test	Expected	Got	
✔	`Drink shot = {"Tequila", 0.6, 30};` `printf("A %d mL drink of %s is about %.1f standard drinks.\n", shot.volume, shot.beverage, shot.volume * shot.abv * 0.0789);`	`A 30 mL drink of Tequila is about 1.4 standard drinks.`	`A 30 mL drink of Tequila is about 1.4 standard drinks.`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **11**

Correct

Mark 1.00 out of 1.00

Given the declaration of `Vector3d` below, write two functions:

- a function `Vector3d vector(int x, int y, int z)` that returns a Vector3d representing the vector (x, y, z)
- a function `Vector3d vectorCrossProduct(Vector3d a, Vector3d b)` that returns the cross product of vectors $\vec{a} = [a_x, a_y, a_z]$ and $\vec{b} = [b_x, b_y, b_z]$, which is defined as the vector $(a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x)$

```
typedef struct {
    int x;
    int y;
    int z;
} Vector3d;
```

For example:

Test	Result

`Vector3d v1 = vector(1, 2, 3);`
`Vector3d v2 = vector(-4, -6, 2);`
`Vector3d v3 = vectorCrossProduct(v1, v2);`
`printf("(%d, %d, %d)\n", v3.x, v3.y, v3.z);`
	`(22, -14, 2)`

Answer: (penalty regime: 0, 10, 20, ... %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
Vector3d vector(int x, int y, int z)
{
    Vector3d createdvector;
    createdvector.x = x;
    createdvector.y = y;
    createdvector.z = z;
    return createdvector;

}



Vector3d vectorCrossProduct(Vector3d a, Vector3d b)
{
    Vector3d crossresult;
    crossresult.x = ((a.y * b.z) - (a.z * b.y));
    crossresult.y = ((a.z * b.x) - (a.x * b.z));
    crossresult.z = ((a.x * b.y) - (a.y * b.x));
    return crossresult;




}
```

	Test	Expected	Got	

✔	`Vector3d v1 = vector(1, 2, 3);`
	`Vector3d v2 = vector(-4, -6, 2);`
	`Vector3d v3 = vectorCrossProduct(v1, v2);`
	`printf("(%d, %d, %d)\n", v3.x, v3.y, v3.z);`
		`(22, -14, 2)`	`(22, -14, 2)`	✔

	Test	Expected	Got	

✔	`Vector3d v1 = vector(2, 0, 0);`
	`Vector3d v2 = vector(0, 3, 0);`
	`Vector3d v3 = vectorCrossProduct(v1, v2);`
	`printf("(%d, %d, %d)\n", v3.x, v3.y, v3.z);`
		`(0, 0, 6)`	`(0, 0, 6)`	✔

✔	`Vector3d v1 = vector(0, -1234, 2174);`
	`Vector3d v2 = vector(-987, 5678, 321);`
	`Vector3d v3 = vectorCrossProduct(v1, v2);`
	`printf("(%d, %d, %d)\n", v3.x, v3.y, v3.z);`
		`(-12740086, -2145738, -1217958)`	`(-12740086, -2145738, -1217958)`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **12**

Correct

Mark 1.00 out of 1.00

In a certain program, the structure for representing a task is defined as:

```
typedef struct {
    char* description;
    float duration; // hours
    int priority;
} Task;
```

Given this declaration for `Task`, write two functions `newTask` and `freeTask` with the signatures:

```
Task* newTask(char* description, float duration, int priority);
void freeTask(Task* task);
```

The `newTask` function returns a pointer to a newly allocated Task on the heap, with the `description`, `duration`, and `priority` fields set appropriately. Here, the `description` field must be a dynamically allocated copy of the corresponding string parameter `description`. All memory (de)allocation must be performed using `malloc` and `free`, and each memory allocation should use the exact required block size—no larger or smaller. You do not need to deal with the possibility that *malloc* fails.

The `freeTask` function frees all memory that was allocated by `newTask` when creating the `Task` currently pointed to by the parameter.

For example:

Test	Result

`Task* task = newTask("Studying for ENCE260", 2.5f, 1);`
`printf("Task \'%s\' (priority %d) takes %.1f hours.\n", task->description, task->priority, task->duration);`
`freeTask(task);`
	`Task 'Studying for ENCE260' (priority 1) takes 2.5 hours.`

Answer: (penalty regime: 0, 10, 20, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
Task* newTask(char* description, float duration, int priority)
{
    Task* data1 = malloc(sizeof(Task));
    char* data2 = malloc((strlen(description) + 1) * sizeof(char));
    data1->description = data2;
    data1->duration = duration;
    data1->priority = priority;
    strncpy((data1->description), description, (strlen(description) * sizeof(char)));
    data1->description[strlen(description)] = '\0';
    return data1;
}

void freeTask(Task* task)
{
    free(task->description);
    free(task);
}
```

	Test	Expected	Got	

✔	`Task* task = newTask("Studying for ENCE260", 2.5f, 1);`
	`printf("Task \'%s\' (priority %d) takes %.1f hours.\n", task->description, task->priority, task->duration);`
	`freeTask(task);`
		`Task 'Studying for ENCE260' (priority 1) takes 2.5 hours.`	`Task 'Studying for ENCE260' (priority 1) takes 2.5 hours.`	✔

	Test	Expected	Got	

✔	`char desc[] = "Making dinner";`
	`Task* task = newTask(desc, 1.0f, 2);`
	`if (task->description == desc) {`
	`  puts("Test feedback: 'description' has not been copied!");`
	`}`
	`freeTask(task);`
	`__check_mem();`
		`Test feedback: total 30 bytes freed`	`Test feedback: total 30 bytes freed`	✔

✔	`Task* task1 = newTask("Studying for ENCE260", 2.5f, 1);`
	`Task* task2 = newTask("Making dinner", 1.0f, 2);`
	`freeTask(task2);`
	`__check_mem();`
	`freeTask(task1);`
	`__check_mem();`
		`Test feedback: total 30 bytes freed`
		`Test feedback: total 67 bytes freed`
			`Test feedback: total 30 bytes freed`
			`Test feedback: total 67 bytes freed`
				✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **13**

Correct

Mark 1.00 out of 1.00

You are to write the declaration of a type `QueueElement`, which contains a student's username and a pointer to the next `QueueElement` in a queue (i.e. a linked list), e.g. of students waiting for a tutor. You are also to write (after the declaration) a function `appendToQueue` which takes a pointer to a queue (i.e. to the first element of the queue, or NULL if the queue is empty) and a pointer to a `QueueElement` to add to the queue. The function should append the given element on the end of the queue (i.e. the end of the linked list) and return a pointer to the head of the queue. If your code is inserted into the following program at the place marked "Your answer is inserted here", the program should compile and run correctly, generating the output shown below it.

```
#include <stdio.h>

// ***************************
//
// Your answer is inserted here
//
// ***************************

// Print the elements in the given queue of users preceded by the word
// 'Queue' and the supplied message.
void printQueue(QueueElement* queue, const char* message)
{
    QueueElement* current = queue;
    printf("Queue %s:", message);

    while (current != NULL) {
        printf(" %s", current->username);
        current = current->next;
    }
    printf("\n");
}


// Simple test of the student queue
int main(void)
{
    QueueElement* queue = NULL;
    QueueElement stud1 = {"abc24", NULL};
    QueueElement stud2 = {"pqr33", NULL};

    printQueue(queue, "at start");
    queue = appendToQueue(queue, &stud1);
    printQueue(queue, "after appending abc24");
    queue = appendToQueue(queue, &stud2);
    printQueue(queue, "after appending pqr33");
}
```

The output from the above program, when completed correctly, should be

```
Queue at start:
Queue after appending abc24: abc24
Queue after appending pqr33: abc24 pqr33
```

Paste **only** your declaration of the QueueElement type and the appendToQueue function into the answer box.

For example:

Test	Result

`// Testing with the same main program as`
`// given above.`
`#include "decl.h" // provides printQueue`
 	
`// Simple test of the student queue`
`int main(void)`
`{`
`  QueueElement* queue = NULL;`
`  QueueElement stud1 = {"abc24", NULL};`
`  QueueElement stud2 = {"pqr33", NULL};`
 	
`  printQueue(queue, "at start");`
`  queue = appendToQueue(queue, &stud1);`
`  printQueue(queue, "after appending abc24");`
`  queue = appendToQueue(queue, &stud2);`
`  printQueue(queue, "after appending pqr33");`
`}`
	`Queue at start:`
	`Queue after appending abc24: abc24`
	`Queue after appending pqr33: abc24 pqr33`

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
typedef struct queue_struct QueueElement;


struct queue_struct {
    char* username;
    QueueElement* next;

};



QueueElement* appendToQueue(QueueElement* firstpointer, QueueElement* addedelement)
{
    if (firstpointer == NULL) {
        firstpointer = addedelement;
    } else {
        QueueElement* first = firstpointer;
        while (first->next != NULL) {
            first = first->next;
        }
        first->next = addedelement;
    }
    return firstpointer;

}
```

	Test	Expected	Got	

✔	`// Testing with the same main program as`
	`// given above.`
	`#include "decl.h" // provides printQueue`
	
	`// Simple test of the student queue`
	`int main(void)`
	`{`
	`  QueueElement* queue = NULL;`
	`  QueueElement stud1 = {"abc24", NULL};`
	`  QueueElement stud2 = {"pqr33", NULL};`
	
	`  printQueue(queue, "at start");`
	`  queue = appendToQueue(queue, &stud1);`
	`  printQueue(queue, "after appending abc24");`
	`  queue = appendToQueue(queue, &stud2);`
	`  printQueue(queue, "after appending pqr33");`
	`}`
		`Queue at start:`
		`Queue after appending abc24: abc24`
		`Queue after appending pqr33: abc24 pqr33`
			`Queue at start:`
			`Queue after appending abc24: abc24`
			`Queue after appending pqr33: abc24 pqr33`
				✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **14**

Not answered

Mark 0.00 out of 1.00

Write a function with signature `char** split(const char* s)` that takes a string *s* and returns a pointer to a new array of strings, allocated on the heap, containing, in order, all the "tokens" from *s*, where a token is defined as a continuous sequence of non-space characters. [This function corresponds roughly to the Python `str.split()` method.] The returned array must be terminated by a NULL pointer. Tokens may be separated by multiple space characters and there may be extra spaces at the start and end of the string, which should be ignored.

This question is checked using valgrind.

For example:

Test	Result

`char** words = split("He said 'hello' to me!");`
`int i = 0;`
`while (words[i] != NULL) {`
`  puts(words[i]);`
`  free(words[i]);`
`  i += 1;`
`}`
`free(words);`
	`He`
	`said`
	`'hello'`
	`to`
	`me!`

Answer: (penalty regime: 0, 10, ... %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

◄ Lab question drill quiz

Jump to...

Lecture Note PDFs ►