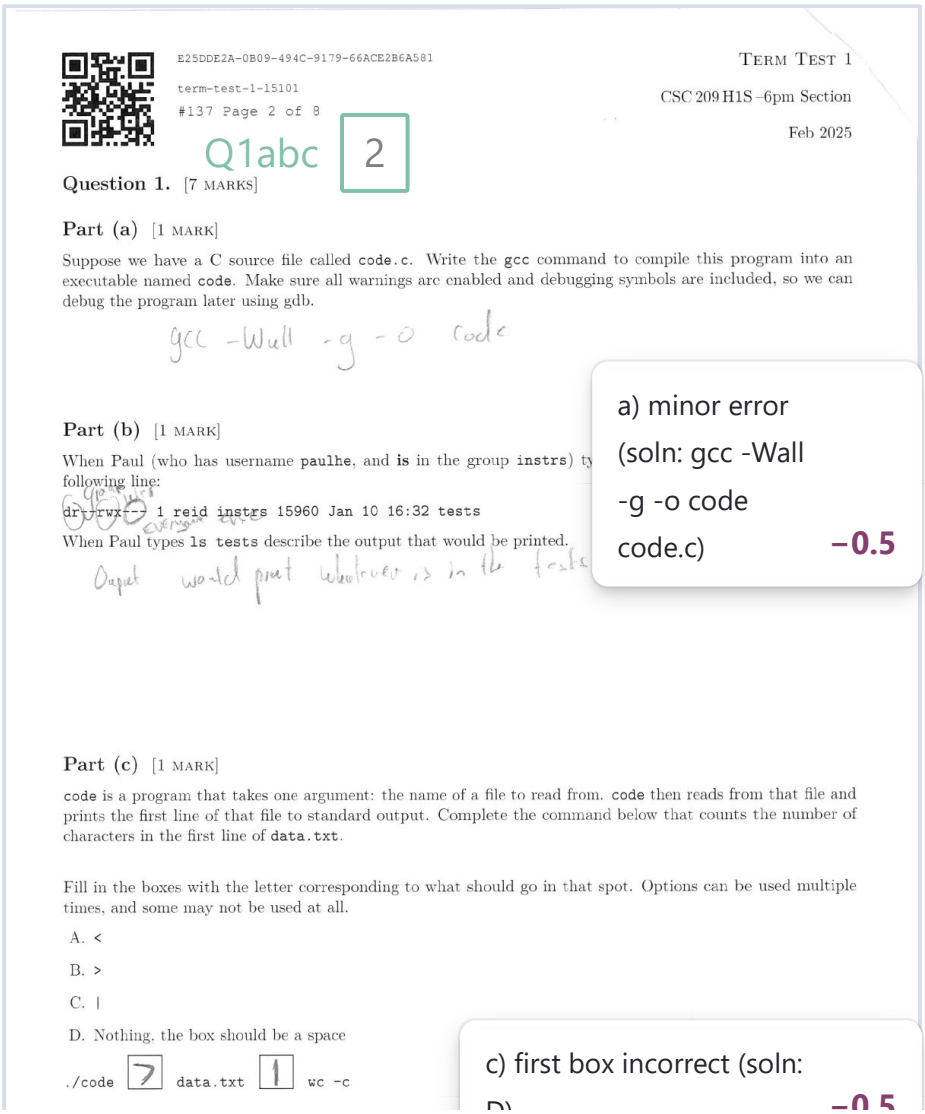
Term Test 1 - L5101

My score

75% (21/28)



E25DDE2A-0B09-494C-9179-66ACE2B6A581

term-test-1-15101

#137 Page 2 of 8

Q1abc 2

TERM TEST 1

CSC 209 H1S –6pm Section

Feb 2025

Question 1. [7 MARKS]

Part (a) [1 MARK]

Suppose we have a C source file called `code.c`. Write the `gcc` command to compile this program into an executable named `code`. Make sure all warnings are enabled and debugging symbols are included, so we can debug the program later using gdb.

gcc -Wall -g -o code

a) minor error (soln: gcc -Wall -g -o code code.c) −0.5

Part (b) [1 MARK]

When Paul (who has username `paulhe`, and **is** in the group `instrs`) ty
following line:

```
dr--rwx--- 1 reid instrs 15960 Jan 10 16:32 tests
```

When Paul types `ls tests` describe the output that would be printed.

Output would print whatever is in the tests

Part (c) [1 MARK]

`code` is a program that takes one argument: the name of a file to read from. `code` then reads from that file and prints the first line of that file to standard output. Complete the command below that counts the number of characters in the first line of `data.txt`.

Fill in the boxes with the letter corresponding to what should go in that spot. Options can be used multiple times, and some may not be used at all.

A. <

B. >

C. |

D. Nothing, the box should be a space

./code [7] data.txt [1] wc -c

c) first box incorrect (soln: D) −0.5

D) 0.5

Q1de 3.5

Part (d) [2 marks] Given the declarations below, select all of the expressions that are equivalent to `arr[i]`:

```
int i; // Assume i is initialized to some value
int arr[5];
```

- [x] `*(arr + i)`
- [ ] `*arr + i`
- [x] `*(arr + (i * sizeof(int)))`
- [ ] `*arr[i]`

d) _ (Box should not be checked) −0.5

Part (e) [2 marks] Select all of the choices that are valid ways of **safely** copying as much of the unknown string `source` as possible into `dest` so that `dest` is a valid string afterwards:

```
char dest[CAPACITY];
char *source; // assume source is initialized to a valid string
```

- [ ] `strncpy(dest, source, CAPACITY - 1);`
- [ ] `strncpy(dest, source, strlen(source) + 1);`
- [ ] `strncpy(dest, source, CAPACITY - 1); dest[CAPACITY] = '\0';`
- [ ] `strncpy(dest, source, CAPACITY);      dest[CAPACITY] = '\0';`
- [x] `strncpy(dest, source, CAPACITY - 1); dest[CAPACITY - 1] = '\0';`
- [x] `strncpy(dest, source, CAPACITY);      dest[CAPACITY - 1] = '\0';`

Q2 4

Question 2. [4 MARKS]

Write the most appropriate **type** in each blank space. Consider each subquestion separately. Assume that the variables are assigned appropriate values such that the code would compile and run correctly.

a)
```
char **s;

_____ p = s[1];

_____ q = &s;
```

b)
```
int *x;

_____ p = malloc(10 * sizeof(int));

_____ q = x[p[0]];
```

c)
```
double **x;

_____ p = x[12];

_____ q = *(p + 1);
```

d)
```
struct foo {
    int num;
};

void reset(_____ f) {
    f->num = 0;
}

int main() {

    _____ f;
    reset(&f);
    return 0;
}
```

All correct 4

Q3 8

Question 3. [8 MARKS] Consider the program below that compiles and runs correctly.

Fill in the memory diagram to show the current state of the program exactly before the return statement on **line 8** is executed. If there are uninitialized blocks of memory at that point in the program, write their values as ???. Clearly label and separate the stack frames.

```
1  int *sum(int *a, int *b, int n) {
2
3      int *result = malloc(n * sizeof(int));
4
5      for (int i = 0; i < n; i++) {
6          result[i] += a[i] + b[i];
7      }
8      return result;
```

Section	Address	Value	Label
Read-only	0x104		
	0x108		
	0x10c		
Heap	0x240	31	
	0x244	42	
	0x248		

```
}

int main() {
    int arr1[2] = {11, 12};
    int arr2[2] = {20, 30};

    int *ptr = sum(arr1, arr2, 2);

    free(ptr);
    return 0;
}
```

Stack Main

0x444	11	arr1
0x448	12	
0x44c	20	arr2
0x450	30	
0x454	0x240	ptr
0x458		
0x45c		

Sum

0x460	0x444	a
0x464		
0x468	0x44c	b
0x46c		
0x470	2	n
0x474	0x240	result
0x478		
0x47c	~~0~~ 2	i
0x480		
0x484		

Correct! **8**

56A72837-B2E8-4DF4-A7BC-BE9332111329

2.5

Term-Test-1-15101

#137 Page 6 of 8

TERM TEST 1

CSC 209 H1S –6pm Section

Feb 2025

Question 4. [3 MARKS] Given the following constants and struct definition, complete the code below so that it compiles and runs without error, and behaves according to the comments. The output when running the program should be `Name: Adaine Class: Wizard`.

```
#define NUM 5
#define MAX_LEN 32

struct role {
    char *name;
    char class[MAX_LEN];
};

// Initialize the name field with new_name and the class field with new_class
void set_role(struct role *r, char *new_name, char *new_class) {
    r->name = malloc ( sizeof (char) * strlen(new_name)+1);
    strcpy(r->name, new_name, strlen(new_name));
    r->name[strlen(new_name)] = '\0';
    strncpy(r->class, new_class, MAX_len-1);
}

int main() {
    // dynamically allocate space for an array of size NUM
    struct role *roles = malloc(NUM * sizeof(struct role));

    // set the 0th element of roles

    set_role(roles, "Adaine", "Wizard");

    // print the fields of the 0th element of roles

    printf("Name: %s Class: %s\n", roles->name, r
    return 0;
}
```

Minor issue in set_role (note allocating new space for name is ok, but not necessary) **−0.5**

Q5 1

TERM TEST 1

CSC 209 H1S –6pm Section

Feb 2025

95489C80-E968-4D8A-95DA-16DD586B400F

term-test-1-15101

#137 Page 7 of 8

Question 5. [6 MARKS]

Complete the function below that takes a string as an argument and replaces all adjacent duplicated characters with a single character. The function should return a string that is the minimum size required to hold the resulting string. Assume `str` is a valid string.

Example: `remove_duplicates("aabcccbd")` returns `"abcbd"`

```
char *remove_duplicates(char *str) {
    int length = strlen(str);
char check[strlen(str)+1];   # space for null terminator
check[0] = '\0'; ... check[1] = '\0'; ... check[2] = '\0', etc to check[length] = '\0';
    int increment = 0;
for (int i=0, i<length, i++) {
    if (i == 0) {
        check[0] = str[0] }
    else if ( check[increment] != str[i] ) {
        check[increment+1] = str[i];
        increment ++;
    }
}
if (increment == length) {
    check[length +1] = '\0';   # [illegible]
}
```

ab

Code tracing & debugging

increment	i	str
0 a	0	
0 a	1	
1 ab	2	
2 c	3	

Logic: # my idea was queue of all [illegible] checking if the [illegible] the end of the queue [illegible] appending to the queue respectively to deal with adjacency.

Please see reference solution. There are too many problems with this answer. **0**

D8279B60-2A55-40C3-922C-E7DFB7241FB3

term-test-1-15101

#137 Page 8 of 8

"Blank" page for rough work or answers.

Total Marks = 28