Week 08 Weekly Test Sample Answers

Test Conditions

These questions must be completed under self-administered exam-like conditions. You must time the test yourself and ensure you comply with the conditions below.

- You may complete this test in CSE labs or elsewhere using your own machine.
- You may complete this test at any time before **Wed Nov 11 21:00:00 2020**.
- Weekly tests are designed to act like a past paper - to give you an idea of how well you are progressing in the course, and what you need to work on. Many of the questions in weekly tests are from past final exams.
- Once the first hour has finished, you must submit all questions you've worked on.
- You should then take note of how far you got, which parts you didn't understand.
- You may choose then to keep working and submit test question anytime up to Wed Nov 11 21:00:00 2020
- However the maximum mark for any question you submit after the first hour will be 50%

You may access this **language documentation** while attempting this test:

- C quick reference
- MIPS quick reference

You may also access manual entries (the `man` command).

Any violation of the test conditions will results in a mark of zero for the entire weekly test component.

Set up for the test by creating a new directory called `test08`, changing to this directory, and fetching the provided code by running these commands:

```
$ mkdir test08
$ cd test08
$ 1521 fetch test08
```

Or, if you're not working on CSE, you can download the provided code as a zip file or a tar file.

WEEKLY TEST QUESTION:

Check a File for non-ASCII Bytes

We need to check whether files contain non-ASCII bytes.

Write a C program, `non_ascii.c`, which takes one argument, a filename.

It should print one line of output.

If the file contains a non-ASCII byte, `non_ascii.c` should print the location of the first non-ASCII byte. Use the same format as the example below.

If the file contains no non-ASCII byte `non_ascii.c` should print a message indicating this. Again use the same format as the example below.

Assume a byte is non-ASCII if it contains a value between 128..255 inclusive.

```
$ dcc non_ascii.c -o non_ascii
$ echo hello world >file1
$ ./non_ascii file1
file1 is all ASCII
$  echo -e 'hello\xBAworld' >file2
$ ./non_ascii file2
file2: byte 5 is non-ASCII
$ ./non_ascii non_ascii.c
non_ascii.c is all ASCII
$  echo -e '\x80\x81' >file3
file2: byte 0 is non-ASCII
```

HINT:

Use *fopen* to open the file, and *fgetc* to read the file.

NOTE:

No error checking is required.

Your program can assume it is always given the name of a file.

Your solution must be in C only.

You are not permitted to run external programs. You are not permitted to use system, popen, posix_spawn, fork or exec.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 1521 autotest non_ascii
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1521 test08_non_ascii non_ascii.c
```

Sample solution for `non_ascii.c`

```c
// Print position of first non-ASCII byte in file

#include <stdio.h>
#include <stdlib.h>

void process_file(char *pathname);

int main(int argc, char *argv[]) {
    for (int arg = 1; arg < argc; arg++) {
        process_file(argv[arg]);
    }
    return 0;
}

void process_file(char *pathname) {
    FILE *stream = fopen(pathname, "r");
    if (stream == NULL) {
        perror(pathname);
        exit(1);
    }

    ssize_t byte_position = 0;
    int byte;
    while ((byte = fgetc(stream)) != EOF) {
        if (byte > 127) {
            printf("%s: byte %zd is non-ASCII\n",  pathname, byte_position);
            return;
        }
        byte_position++;
    }

    fclose(stream);
    printf("%s is all ASCII\n", pathname);
}
```

WEEKLY TEST QUESTION:

Compare the Bytes of Two Files

We need to check whether 2 files contain identical bytes

Write a C program, `compare_file.c`, which takes two arguments, both filenames.

`compare_file.c` should print one line of output.

If the two files are different `compare_file.c` should print the location of the first byte whether they differ. Use the same format as the example below.

If the one file is shorter than the other but the bytes it contains are identical to the other file, `compare_file.c` should print a message indicating this. Use the same format as the example below.

If the 2 files contain exactly the same bytes `compare_file.c` should print a message indicating this. Again use the same format as the example below.

```
$ dcc compare_file.c -o compare_file
$ echo hello world >file1
$ echo hello world >file2
$ ./compare_file file1 file2
Files are identical
$ echo -n hello >file3
$ ./compare_file file1 file3
EOF on file3
$ echo help me >file4
$ ./compare_file file1 file4
Files differ at byte 3
```

> **NOTE:**
>
> No error checking is required.
> Your program can assume it is always given the names of 2 files.
>
> Your solution must be in C only.
>
> You are not permitted to run external programs. You are not permitted to use system, popen, posix_spawn, fork or exec.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 1521 autotest compare_file
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1521 test08_compare_file compare_file.c
```

Sample solution for `compare_file.c`

```c
// Print position of first non-ASCII byte in file

#include <stdio.h>
#include <stdlib.h>

int compare_files(char *pathname1, char *pathname2);

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <file1> <file2>\n", argv[0]);
        return 1;
    }

    return compare_files(argv[1], argv[2]);
}

int compare_files(char *pathname1, char *pathname2) {
    FILE *stream1 = fopen(pathname1, "r");
    if (stream1 == NULL) {
        perror(pathname1);
        return 1;
    }

    FILE *stream2 = fopen(pathname2, "r");
    if (stream2 == NULL) {
        perror(pathname2);
        return 1;
    }

    ssize_t byte_position = 0;
    int byte1, byte2;
    while (1) {
        byte1 = fgetc(stream1);
        byte2 = fgetc(stream2);
        if (byte1 != byte2 || byte1 == EOF) {
            break;
        }
        byte_position++;
    }

    if (byte1 == byte2) {
        printf("Files are identical\n");
    } else if (byte1 == EOF) {
        printf("EOF on %s\n", pathname1);
    } else if (byte2 == EOF) {
        printf("EOF on %s\n", pathname2);
    } else {
        printf("Files differ at byte %zd\n", byte_position);
    }

    fclose(stream1);
    fclose(stream2);
    return 0;
}
```

WEEKLY TEST QUESTION:

Delete non-ASCII characters from a file

We need to remove the non-ASCII bytes from files.

Write a C program, `leave_only_ascii.c`, which takes one argument, a filename.

`leave_only_ascii.c` should remove all non-ASCII bytes from the file.

After it is run the file should contain only a ASCII bytes.

It should print nothing on stdout. It should only change the file.

Assume a byte is non-ASCII if it contains a value between 128..255 inclusive.

```
$ dcc leave_only_ascii.c -o leave_only_ascii
$ echo -e 'hello\xBAworld' >file2
$ ls -l file2
-rw-r--r-- 1 z5555555 z5555555 12 Nov  8 08:18 file2
$ ./non_ascii file2
file2: byte 5 is non-ASCII
$ ./leave_only_ascii file2
$ ls -l file2
-rw-r--r-- 1 z5555555 z5555555 11 Nov  8 08:18 file2
$ ./non_ascii file2
file2 is all ASCII
```

NOTE:

No error checking is required.
Your program can assume it is always given the name of a file.

You are permitted to create another (temporary) file in the current directory.

Your solution must be in C only.

You are not permitted to run external programs. You are not permitted to use system, popen, posix_spawn, fork or exec.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 1521 autotest leave_only_ascii
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1521 test08_leave_only_ascii leave_only_ascii.c
```

Sample solution for `leave_only_ascii.c`

```c
// Print on-ASCII bytes from file

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

void process_file(char *pathname);

int main(int argc, char *argv[]) {
    for (int arg = 1; arg < argc; arg++) {
        process_file(argv[arg]);
    }
    return 0;
}

void process_file(char *pathname) {
    FILE *in = fopen(pathname, "r");
    if (in == NULL) {
        perror(pathname);
        exit(1);
    }
    char *suffix = ".tmp.leave_only_ascii";
    char temporary_pathname[strlen(pathname) + strlen(suffix)  + 1];
    strcpy(temporary_pathname, pathname);
    strcat(temporary_pathname, suffix);

    FILE *out = fopen(temporary_pathname, "w");
    if (out == NULL) {
        perror(pathname);
        exit(1);
    }

    int byte;
    while ((byte = fgetc(in)) != EOF) {
        if (byte <= 127) {
            fputc(byte, out);
        }
    }
    fclose(in);
    fclose(out);

    // doesn't prevserve file permissions
    if (rename(temporary_pathname, pathname) != 0) {
        perror("rename");
        exit(1);
    }
}
```

Submission

When you are finished each exercise make sure you submit your work by running **give**.

You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Wed Nov 11 21:00:00 2020** to complete this test.

Automarking will be run by the lecturer several days after the submission deadline for the test, using test cases that you haven't seen: different to the test cases `autotest` runs for you.

(Hint: do your own testing as well as running `autotest`)

Test Marks

After automarking is run by the lecturer you can view it here the resulting mark will also be available via via give's web interface or by running this command on a CSE machine:

```
$ 1521 classrun -sturec
```

The test exercises for each week are worth in total 1 marks.

The best 6 of your 8 test marks for weeks 3-10 will be summed to give you a mark out of 9.