This course has already ended.

« Example Exam Course materials Exam »
ELEC-A7100 / Example Exam / 1 Example Exam

Example Exam¶

In the exam, there are five questions on the following topics:

1. **Arrays in C**: You are required to implement a single function that performs the specified operation.
2. **Strings in C**: You are required to implement a single function that performs the specified string operation.
3. **Bit operations in C**: You are required to implement a single function that performs the specified bit operations.
4. **Fix code**: You are required to fix errors in the provided code.
5. **Program**: You are required to write a program that has the specified functionality.

Getting Started

1. Each question is shown in its exercise box. **By the end of the description, you can find a link to download** *Visual Studio Code* **workspace folder along with the source code template for each question.**
   - **Note that you need to download different files for each question!**
2. You can download the ZIP file as you did with the exercise modules.
   - If you are using Windows Subsystem for Linux (WSL), Linux or MacOS based system, you can use `wget` tool. `download url` can be copied by right clicking the download link and selecting `copy link`.

     ```
     wget --no-check-certificate <download url>
     ```

   - After downloading the template, you need to unzip its content. If you are using Windows Subsystem for Linux (WSL), Linux or MacOS based system, you can use `unzip` tool.
   - The zip file names end with a number. These are used for personalization of the exam, and does not signify anything on your side.
3. **You can open the *Visual Studio Code* workspace.**
   - Navigate to the directory (folder) you extracted the template.
   - If you are using command-line, enter the following command:

     ```
     code exam.code-workspace
     ```

   - Otherwise, you can do the same by clicking `File - Open Workspace ...`.
4. **Read the comments in the template.**

Testing

- The provided workspace has the following `Run Tasks` pre-configured. You can access them by clicking `Terminal - Run Task...`.
  - **Build**: builds the source file `main.c` and creates executable `main.out`.
  - **Clean**: deletes the executable file `main.out`.
  - **Valgrind**: runs `valgrind` with `main.out`.
- The provided workspace has a launch configuration pre-configured so that you can use the debugger right away.
- The template files **do not** have any tests for your implementations. You are responsible for creating the tests for the first 3 questions if you need validation.
  - Your test code will not be evaluated (except question 5: **Program**).

Submission

- You can submit your solutions as you did in the exercises.
- **You should use the correct submission box of the solution.**
- *Your file should have a valid* `main` *function.*
- The number of submissions is not limited, but it is faster and easier to develop and test on your local environment.
- After submission, you can see `compiler` and `valgrind` outputs below the question description.
- If there are `compiler warnings` or memory leaks detected by `valgrind`, you will see some penalties. Their importance depends on the question. In some questions, `valgrind` errors are ignored, and in some `compiler warnings` originating from `main` or `test` functions are ignored.

Evaluation

- We will manually grade your solutions.
- You will be graded based on your last submissions.
- Your coding style might affect your grade as it is in the programming task. Therefore, pay attention when naming your structures, variables and functions.

Points 0 / 0 My submissions 1 Deadline Tuesday, 31 August 2021, 23:59
To be submitted alone

This course has been archived (Saturday, 31 December 2022, 20:00).

Strings in C

Create `mystrcat(char *dest, const char *src)` function and a `main` function that tests its functionality. The function concatenates string `src` after string `dest`. The function does not allocate memory, but it assumes that the calling function arranges the memory somehow (as with the original `strcat` function). The function returns pointer to the beginning of the string. In this task you must not use the `strcat` function that is defined in `string.h` header.

You can download the template `ZIP file from here.`

main.c

Choose File No file chosen

Submit

Points 0 / 0 My submissions 1 Deadline Tuesday, 31 August 2021, 23:59
To be submitted alone

This course has been archived (Saturday, 31 December 2022, 20:00).

Arrays in C

Create `unsigned int pickmax(unsigned int *numbers)` function and a `main` function that tests its functionality. The function processes table `numbers` that contains unsigned integers. The table ends with number `0`. The function should return *the largest integer in the table*.

You can download the template `ZIP file from here.`

main.c

Choose File No file chosen

Submit

Points 0 / 0 My submissions 1 Deadline Tuesday, 31 August 2021, 23:59
To be submitted alone

This course has been archived (Saturday, 31 December 2022, 20:00).

Bit operations in C

Create `void set_bit(unsigned char *buffer, unsigned int n, int bit)` function and a `main` function that tests its functionality. The function goes through `n` bytes starting from address `buffer`, and sets bit number `bit` in each of the bytes. The most significant bit is numbered `7`, and the least significant bit is numbered `0`.

You can download the template `ZIP file from here.`

main.c

Choose File No file chosen

Submit

Points 0 / 0 My submissions 2 Deadline Tuesday, 31 August 2021, 23:59
To be submitted alone

This course has been archived (Saturday, 31 December 2022, 20:00).

Fix code

The provided functions do not work as specified. Fix functions such that they work as specified, and do not produce compiler warnings or valgrind errors. The following three functions should be fixed:

- **strchr** that searches character `c` from string `str` and returns pointer to it. If the character cannot be found, the function return `NULL`. Function can also match the final nil character, i.e., if you give `\0` as argument `c`, the function returns pointer to the end of the string.
- **createArray** that dynamically allocates memory to store an integer array of `size` numbers, and initializes the array by increasing integers, starting from `0`. The function returns pointer to the beginning of the array. You do not need to worry about releasing the memory: the calling function will take care of it. In this task you can assume that memory allocation always succeeds.
- **addProduct** that adds a new item (of type `struct products`) at the end of dynamically allocated array, and initializes the item as indicated in arguments `newtitle` and `newprice`. The earlier length of the table is indicated by argument `length`. Function returns pointer to the beginning of the extended array. You can assume that memory allocation always succeeds.

Implement `main` function that tests the other functions by at least two different parameter combinations.

You can download the template `ZIP file from here.`

main.c

Choose File No file chosen

Submit

Points 0 / 0 My submissions 2 Deadline Tuesday, 31 August 2021, 23:59
To be submitted alone

This course has been archived (Saturday, 31 December 2022, 20:00).

Program

Design a registration system for an imaginary course. The system lists the names and student numbers of the participating students. You should use and extend the following given data structures in your program:

The `Course` structure lists a number of Student structures in some way, but you can decide yourself how. The implementation could be, for example, a *linked list* or *dynamically allocated array*. Complete the structures by adding missing fields, but the given fields in `Student` structure must not be modified.

Implement the following functions:

1. `void add_student(Course *c, const char *name, const char *ID)` that adds student by name name and student number ID to the student database that is indicated by `Course` structure type pointer `c`.
2. `void remove_student(Course *c, const char *ID)` that removes the given student (ID) from `Student` list `c`. You can assume that an ID is stored in the list at most once.

The implementation must not use more memory than needed for the current student list. On the other hand, it must be able to store also large amounts of students. In other words, you will need to apply dynamic memory management in some way. You can assume that memory allocations always succeed.

You will get two points for successful definition of data structures, and two points for each of the functions. Implement your own main function that tests both functions with at least three different inputs. Successful main function will grant you two points. Altogether there are eight points in this task.

Test your code with `valgrind` as well.

You can download the template `ZIP file from here.`

main.c

Choose File No file chosen

Submit