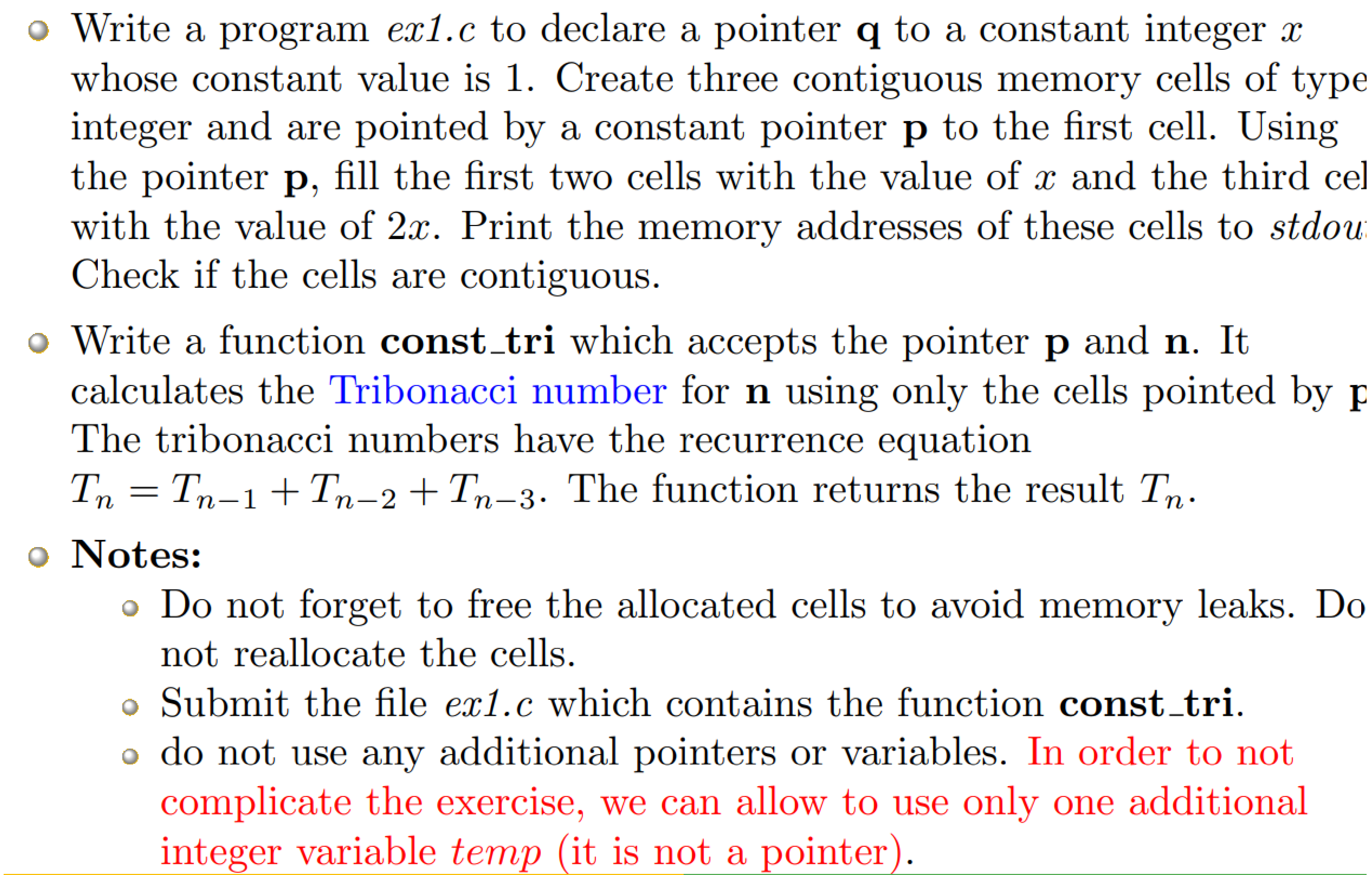
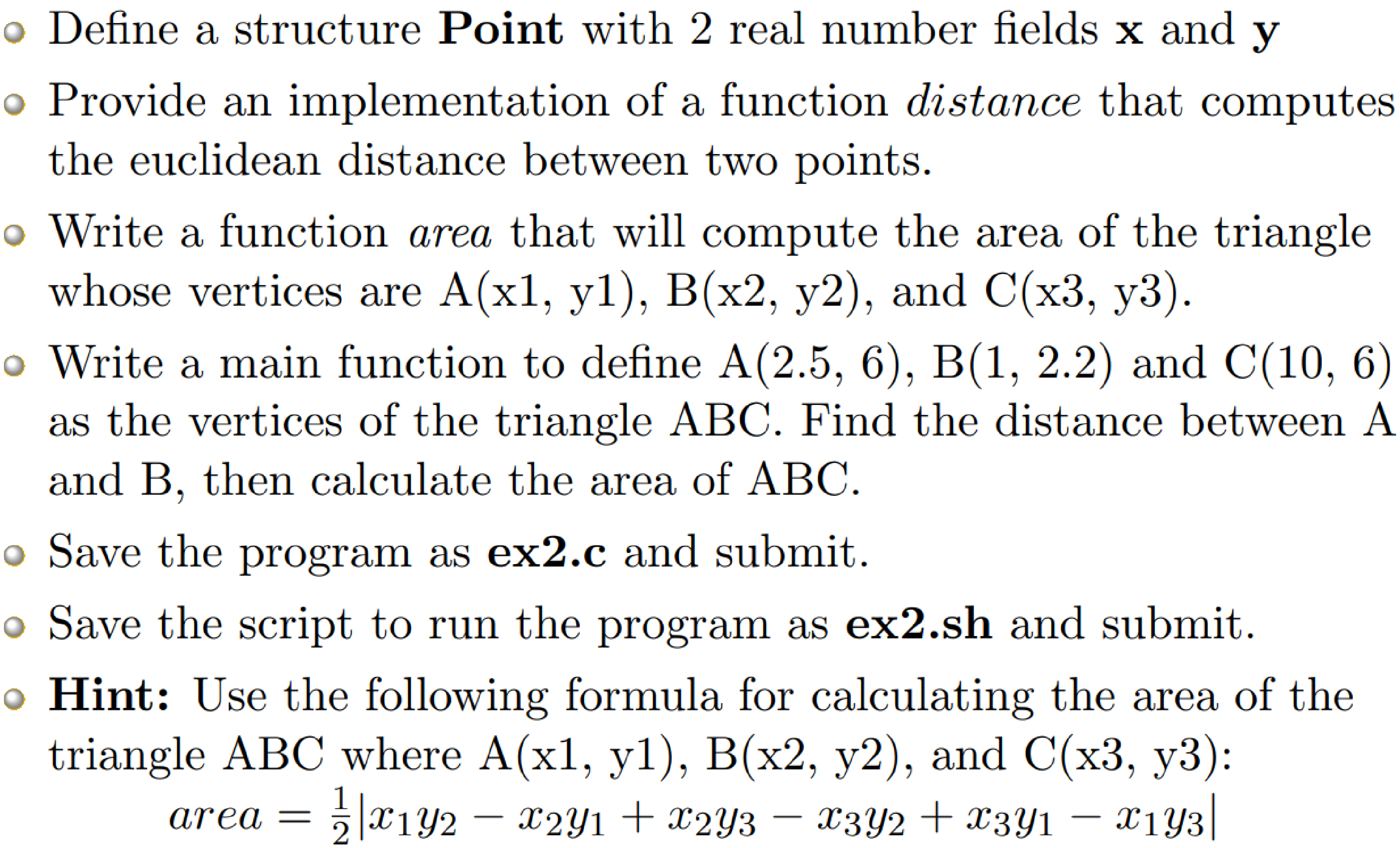
Exercise 1



- Write a program *ex1.c* to declare a pointer **q** to a constant integer x whose constant value is 1. Create three contiguous memory cells of type integer and are pointed by a constant pointer **p** to the first cell. Using the pointer **p**, fill the first two cells with the value of x and the third cel with the value of $2x$. Print the memory addresses of these cells to *stdou* Check if the cells are contiguous.
- Write a function **const_tri** which accepts the pointer **p** and **n**. It calculates the Tribonacci number for **n** using only the cells pointed by **p** The tribonacci numbers have the recurrence equation $T_n = T_{n-1} + T_{n-2} + T_{n-3}$. The function returns the result T_n.
- **Notes:**
  - Do not forget to free the allocated cells to avoid memory leaks. Do not reallocate the cells.
  - Submit the file *ex1.c* which contains the function **const_tri**.
  - do not use any additional pointers or variables. In order to not complicate the exercise, we can allow to use only one additional integer variable *temp* (it is not a pointer).

Exercise 2



- Define a structure **Point** with 2 real number fields **x** and **y**
- Provide an implementation of a function *distance* that computes the euclidean distance between two points.
- Write a function *area* that will compute the area of the triangle whose vertices are A(x1, y1), B(x2, y2), and C(x3, y3).
- Write a main function to define A(2.5, 6), B(1, 2.2) and C(10, 6) as the vertices of the triangle ABC. Find the distance between A and B, then calculate the area of ABC.
- Save the program as **ex2.c** and submit.
- Save the script to run the program as **ex2.sh** and submit.
- **Hint:** Use the following formula for calculating the area of the triangle ABC where A(x1, y1), B(x2, y2), and C(x3, y3):

$$area = \frac{1}{2}|x_1y_2 - x_2y_1 + x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3|$$

Exercise 3 (1/3)

- Files and directories in most operating systems are organized in hierarchical manner. In the Linux-based OS, you have a root directory (/) which does not have a parent directory and contains all other sub-directories and files in the system. In this exercise, you will create a simple system for organizing your files and directories hierarchically using C structures* and pointers.
- Create a `struct File` which represents a file with the fields (**id**: unique number assigned to each file, **name**: name of the file, **size**: current size of the file data, **data**: the actual textual content of the file as a string, **directory**: the directory of type `struct Directory` where the file is in). The structure supports the following operations on files:
  - **overwrite_to_file(struct File* file, const char* str)** which overwrites the file content *file* with the new content *str*.
  - **append_to_file(struct File* file, const char* str)** which appends the new content *str* to the end of the file *file*.
  - **printp_file(struct File* file)** prints to *stdout* the path of file *file*.

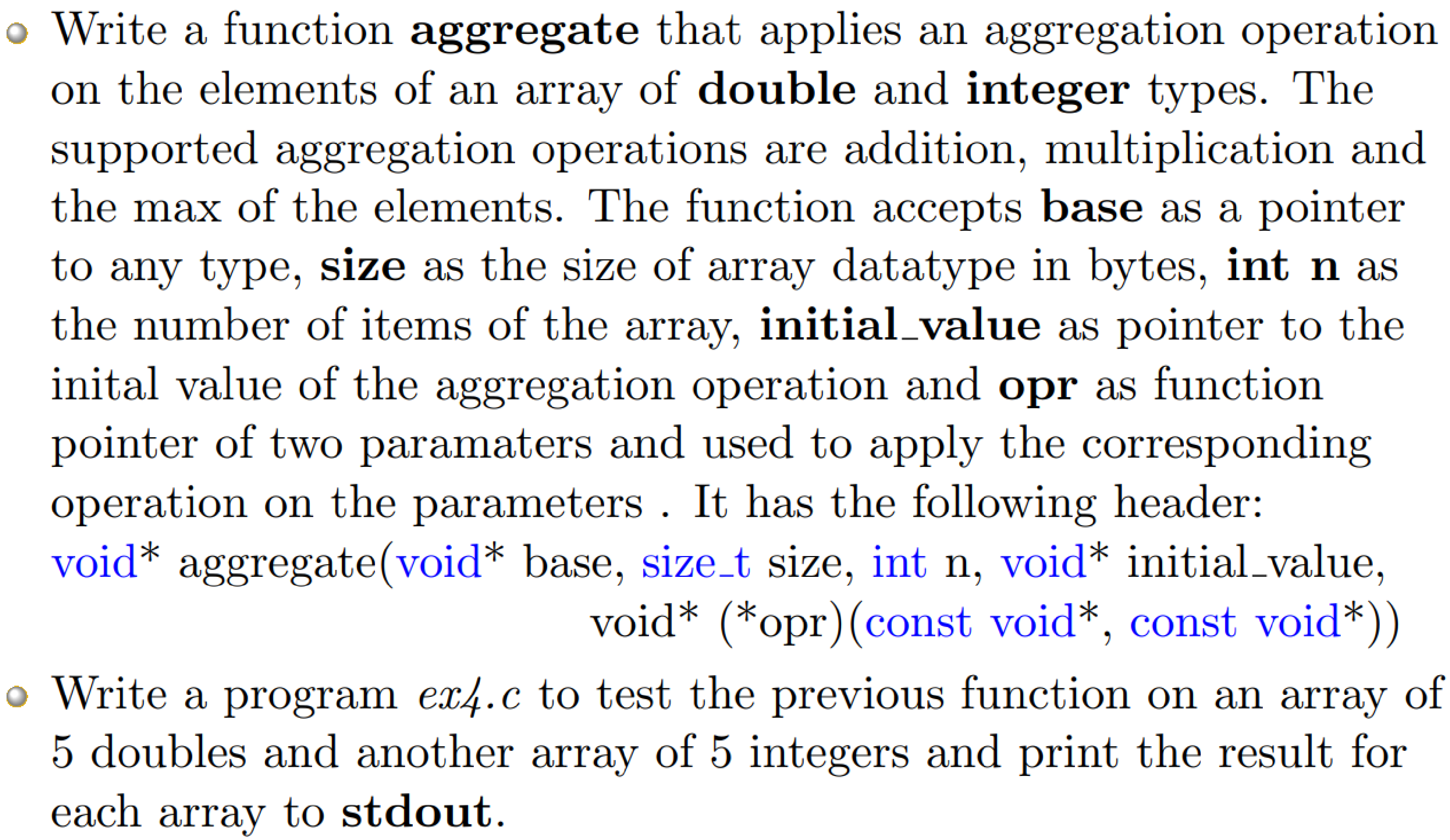
Exercise 3 (2/3)

- Create a `struct Directory` which represents a directory with the fields (**name**: the directory name,**files**: array of files, **directories**: array of sub-directories, **nf**: number of files in the directory, **nd**: number of sub-directories in the current directory, **path**: the absolute path of this directory). It supports the following operations on directories:
  - **add_file(struct File* file, struct Directory* dir)** which adds a new file *file* to the current directory *dir*.
- write a program *ex3.c* contains a *main* function.
  - Create the root directory (/) with two subdirectories **home** and **bin**.
  - Add a file **bash** to the directory **bin**.
  - Add two files ex3_1.c and ex3_2.c to the directory **home**. The file ex3_1.c contains the code: “int printf(const char * format, ...);” And the file ex3_2.c contains the code: “//This is a comment in C language”
  - Add the content “Bourne Again Shell!!” to the file **bash**.

Exercise 3 (3/3)

  - Append the content “int main(){printf(”Hello World!”)}” to the file **ex3_1.c**
- Print the path of all files in the system by calling the function **printp**$_{f}ile$.
- **Notes:**
  - the datatype of nf and nd in `struct Directory` is *unsigned char*. We did not specify the datatype of **id** field. It is up to you to determine the datatype taking into consideration the maximum number of files that the system can hold.
  - the maximum length of the file name is 63 characters.
  - the maximum size of the path is 2048.
  - the maximum size of file data is 1024.
  - The path in this system starts with “/”.
  - size field in `struct File` is the current size of the file data or the length including the null character. Here we do not mean the maximum size of this field.
  - Submit the file *ex3.c* and a supplement script *ex3.sh* to run it.

Exercise 4 (1/2)



- Write a function **aggregate** that applies an aggregation operation on the elements of an array of **double** and **integer** types. The supported aggregation operations are addition, multiplication and the max of the elements. The function accepts **base** as a pointer to any type, **size** as the size of array datatype in bytes, **int n** as the number of items of the array, **initial_value** as pointer to the inital value of the aggregation operation and **opr** as function pointer of two paramaters and used to apply the corresponding operation on the parameters . It has the following header:

```
void* aggregate(void* base, size_t size, int n, void* initial_value,
                                    void* (*opr)(const void*, const void*))
```

- Write a program *ex4.c* to test the previous function on an array of 5 doubles and another array of 5 integers and print the result for each array to **stdout**.

Exercise 4 (2/2)

- **Notes:**
  - The return type of **aggregate** function is void*
  - You cannot change the header of the function **aggregate**.
  - We assume that we have only 2 types of arrays (double and integer arrays).
  - It is up to you to set the values of the arrays but the size of each array should be 5.
  - the initial value of the addition operation is 0 and the initial value of the multiplication operation is 1. For the max operation, you should use the minimum number for the chosen datatype as the initial value.
  - Submit the file *ex4.c*.
- **Hint:** Use *sizeof* to check the specific datatype of the void* pointer.