Exam 1 - programming

Due Dec 23 at 7pm **Points** 30 **Questions** 1
Available Dec 23 at 3pm - Dec 23 at 7pm about 4 hours **Time Limit** None
Allowed Attempts Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	91 minutes	0 out of 30 *

* Some questions not yet graded

ⓘ Correct answers are hidden.

Score for this attempt: **0** out of 30 *
Submitted Dec 23 at 5:05pm
This attempt took 91 minutes.

Question 1 **Not yet graded / 30 pts**

Parity array

In this task a *parity array* type should be implemented. This is an array of integers where even numbers go to the beginning and the odd numbers go to the end of the array.

For example such a 5 element array is initially empty: _ _ _ _ _

After adding 2: 2 _ _ _ _
After adding 7: 2 _ _ _ 7
After adding 1: 2 _ _ 1 7
After adding 9: 2 _ 9 1 7
After adding 4: 2 4 9 1 6

Base task (10 points)

Create a struct named `ParityArray`. This contains a fixed-size array. The size should be given by a preprocessor macro token named `ARRAY_SIZE`. Its value should be 10. Furthermore, the structure contains two integers: `even_idx` and `odd_idx`. These indicate the positions where the next even and odd number is inserted in the array.

Create a function `init()` which is given a `ParityArray` as parameter and initializes it. It sets the value of `even_idx` to 0 and `odd_idx` to `ARRAY_SIZE-1`.

Create a function `insert()` which is given a `ParityArray` and an integer parameter. This function inserts the integer in the array to its correct position and sets the indexes accordingly.

Create a function `print()` which is given a `ParityArray` and prints its values. Make sure to print only the inserted elements. Empty elements shouldn't be printed.

Heap usage (10 points)

Modify the struct so it contains a pointer instead of the array. This pointer will point to an array which is allocated in the heap memory. The size of this array should be given as parameter to `init()` function. Modify `init()` function so it allocates the array in the heap memory. The macro `ARRAY_SIZE` doesn't need to be used anymore, but don't remove it from your source code.

Create a function `destruct()` which is given a `ParityArray` as parameter and deallocates the underlying array.

Modularization (5 points)

Separate your program to multiple translation units. The `init()`, `insert()`, `print()` and `destruct()` function definitions should be replaced to a different source file. Create the corresponding header file which contains the `ParityArray` struct and the function declarations. Don't forget the header-guard idiom.

I/O handling (5 points)

Read a number from the keyboard. Create a `ParityArray` and initialize it with this size. Then read this many integers from the user and insert those numbers to the `ParityArray`.

Create a `dump()` function which writes the array elements to a text file.

parity array.zip (https://canvas.elte.hu/files/2066364/download)

Quiz Score: **0** out of 30