

**Assignment 5**  
**Warehouse13**  
**COP 3514 Program Design**

**Due 07/05/2013**

**Assignment reminders:**

- 1. The assignment must be submitted by the due date**
- 2. The extra day for re-submission is ONLY if you have a minor error in your code and you fix it, and even then I will decide if I accept the re-submission or not.**
- 3. Make sure your code compiles on Circe**
- 4. Make sure your code compiles on Circe**
- 5. Make sure your code compiles on Circe**
- 6. Write and debug the assignment one function at a time.**
- 7. Feel free to ask for help, make sure you have debugging messages in your code before you ask for my help and ask for help in a timely manner.**

Warehouse 13 is a secret service warehouse where supernatural artifacts are being stored. These artifacts are very dangerous and have to be stored and handled properly. Each artifact can be activated and in order to prevent destruction it has to be neutralized. The warehouse has a special design. Every time a new artifact is being stored in the warehouse, the warehouse expands to make room for it. Each time an artifact is neutralized the warehouse destroys the artifact and shrinks.

In this assignment you are to complete the functions that create, show, store, activate and neutralize an artifact.

There are 2 structures defined in the header file: Vault and Artifact.

The Artifact structure has 4 parameters: id, active, hazardLevel and name. The name parameter is a pointer to a character array.

The Vault parameter has 2 parameters: artifact and nextVault. The artifact parameter inside the Vault structure is a pointer to an Artifact structure and the nextVault parameter is a pointer to a Vault structure.

The warehouse is a pointer to a Vault structure initialized to NULL. Each time an artifact is stored, memory is being allocated and it is stored in a Vault. For example, if warehouse is initialized to NULL, after the first artifact is stored it can be accessed as:

**warehouse->artifact**

If a second artifact is stored it can be accessed as:

**warehouse->nextVault->artifact**

If a third artifact is stored it can be accessed as:

**warehouse->nextVault->nextVault->artifact**

etc.

I have created a showWarehouse() function that can be used in the main.c file for debugging purposes. If you store 4 artifacts (as in main.c) and call showWarehouse you will see each Vault with the address of the variable and each artifact with the address and parameters of the artifact variable (Image 1).

```

Vault Address:0x16e6060    --> Vault Address:0x16e60d0    --> Vault Address:0x16e6140    -->
Artifact Address:0x16e6060 --> Artifact Address:0x16e60d0 --> Artifact Address:0x16e6140 -->
Artifact id:1              --> Artifact id:2              --> Artifact id:3              -->
Name:Bruce Lee's Punching Bag--> Name:Einstein-Grant Bridge Device--> Name:Sodom and Gomorrah Salt Mask-->
Hazard Level:10           --> Hazard Level:20           --> Hazard Level:30           -->
Active:0                  --> Active:1                  --> Active:0                  -->

Vault Address:0x16e61b0    -->
Artifact Address:0x16e61b0 -->
Artifact id:4              -->
Name:Thomas Edison's Bioelectric Stagecoach-->
Hazard Level:20           -->
Active:0                  -->

```

Image 1

### Functions to be completed:

The createArtifact() function allocates memory and sets the parameters of variable of type Artifact.

The storeArtifact() function allocates memory for a Vault in the warehouse and stores the Artifact pointer inside the Vault.

The activate() function goes through each Vault inside of the warehouse and sets the active parameter to 1 if the artifact id matches the \_id provided as a function parameter.

The neutralize() function goes through each Vault inside of the warehouse and when it finds the artifact with the same id as the one provided as function parameter it will free the memory allocated for it and shrink the warehouse. For example: If the artifact to be destroyed is in the second Vault then you have to set the nextVault pointer in the first Vault to point to the third Vault and free the memory of the second Vault.

If the ware house has 4 artifacts, after neutralizing artifact with id=2 it will have only 3 artifacts (Image 2)

```

Vault Address:0x16e6060    --> Vault Address:0x16e60d0    --> Vault Address:0x16e6140    -->
Artifact Address:0x16e6060 --> Artifact Address:0x16e60d0 --> Artifact Address:0x16e6140 -->
Artifact id:1              --> Artifact id:2              --> Artifact id:3              -->
Name:Bruce Lee's Punching Bag--> Name:Einstein-Grant Bridge Device--> Name:Sodom and Gomorrah Salt Mask-->
Hazard Level:10           --> Hazard Level:20           --> Hazard Level:30           -->
Active:0                  --> Active:1                  --> Active:0                  -->

Vault Address:0x16e61b0    -->
Artifact Address:0x16e61b0 -->
Artifact id:4              -->
Name:Thomas Edison's Bioelectric Stagecoach-->
Hazard Level:20           -->
Active:0                  -->

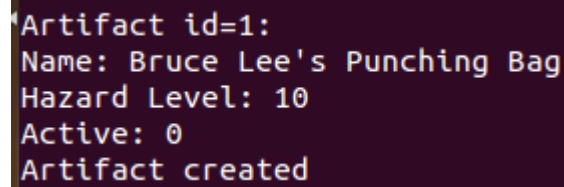
Artifact neutralized

Vault Address:0x16e6060    --> Vault Address:0x16e6140    --> Vault Address:0x16e61b0    -->
Artifact Address:0x16e6060 --> Artifact Address:0x16e6140 --> Artifact Address:0x16e61b0 -->
Artifact id:1              --> Artifact id:3              --> Artifact id:4              -->
Name:Bruce Lee's Punching Bag--> Name:Sodom and Gomorrah Salt Mask--> Name:Thomas Edison's Bioelectric Stagecoach-->
Hazard Level:10           --> Hazard Level:30           --> Hazard Level:20           -->
Active:0                  --> Active:0                  --> Active:0                  -->

```

Image 2

The showArtifact() function prints the parameters of a variable of type Artifact (Image3):

A terminal window with a dark purple background and light blue text. The text displays the output of the showArtifact() function for an artifact with id=1. The output is: Artifact id=1:  
Name: Bruce Lee's Punching Bag  
Hazard Level: 10  
Active: 0  
Artifact created

```
Artifact id=1:
Name: Bruce Lee's Punching Bag
Hazard Level: 10
Active: 0
Artifact created
```

Image 3

The format of the showArtifact() function can be similar to Image 3. This function should be used for debugging purposes.

### EXTRA CREDIT (10 points):

-The sort() function goes through the warehouse Vaults and sorts the artifacts based on their hazardLevel. The artifacts with a high hazardLevel should be in the first Vault (descending order)

### Files:

You are provided with Warehouse13.c , Warehouse13.h , main.c and Makefile. Do not modify Warehouse13.h . Use the main.c file to call and verify the functions in defined in Warehouse13.h . I have provided an example layout of the main file that will give you some guidance on how to call each function and how to check if it is correct or not.

**DO NOT COMMENT OUT THE EXTRA CREDIT FUNCTION EVEN IF YOU ARE NOT GOING TO COMPLETE IT**

### Grading:

Your code will be tested with 9 regular test case inputs (10 points each) and 2 special test case inputs (5 points each)

### Submission:

**Submission should be done only through Blackboard. Submit only Warehouse13.c**

### Compiling and testing:

To compile your file make sure all of the files are in the same directory. Type

**“make --always-make” in the terminal in the directory of the files. If there are no compilation errors a main executable will be created.**

**To test your program run “./main” from the terminal in the directory of the files.**

**Test your functions one by one by calling them in the main function. Use the showWarehouse function to see how the warehouse looks after storing an artifact.**