Task 5. Write a piece of code in a file `task5.c` that does the following:
Run: `module load gcc/latest`

a) Reads in an integer value n, where $0 < n < 100$
b) Allocates dynamically an array of `int`s to store the all integer values between n (including n) and 0 (including 0)
c) Use `qsort` to sort the array in ascending order – from 0 to n
d) Call the function `outputT5` from `output.h` on the array, with `count` being the size of the array, after sorting it.
e) Make sure you free the memory that you allocate in your program

We will test your code with a bunch of integer values n – some positive, some negative, and zero. Make sure you have the right guards in place to only run for legitimate values of n. We will compile your code as follows: `gcc task5.c output.c -o task5.` We will run the code like this: `./task5` n.

1. cd /srv/home/cmiao/ME459Upstream/HW04
2. touch task5.c
3. nano task5.c
4. input the code at bottom.
5. gcc task5.c output.c -o task5 -std=c99
6. ./task5 n

Situation 1: if you input 2 numbers, or the number is not from 1 to 99:

```
[[cmiao@euler HW04]$ ./task5 3 5
 You should input one number from 1 to 99!
 Please run again!
[[cmiao@euler HW04]$ ./task5 -2
 You should input one number from 1 to 99!
 Please run again!
```

Situation 2: if n is good:

```
[[cmiao@euler HW04]$ ./task5 4
[Input your value number 1:23
[Input your value number 2:456
[Input your value number 3:3
[Input your value number 4:35
 The result is:
 3 23 35 456
 [cmiao@euler HW04]$
```

```
#include <stdio.h>
#include <stdlib.h>
#include "output.h"
```

```
int cmp(const void *a, const void *b)
{
   return (*(int *)a - *(int *)b);
}

int main(int argc, char *argv[]){

   int n=atoi(argv[1]), i;
   int *ids=(int*)malloc(sizeof(int)*n);
   if(n<=0 || n>99 || argc!=2){
      printf("The number you just input is be bigger than 0 and smaller then 100!\nPlease run again!\n");
      exit(1);
   }
   else{
   for(i = 0; i < n; i++){
      printf("Input your value number %d:",i+1);
      scanf("%d", &ids[i]);
   }
   qsort(ids, n, sizeof(ids[0]), cmp);
   printf("The result is:\n");
   outputT5(ids, n);
   printf("\n");
   free(ids);
   }
}
```