Lab 3 checkpoint1

Use the make utility to build the project using the command make and then use valgrind to test it by typing “valgrind ./hello”.

```c
#include <stdio.h>
#include <stdlib.h>


int main(int argc, char* argv[])
{
int *p;//Declaration of a pointer variable.

p=(int*)malloc(sizeof(int)*100); //First "bookend" allocates space

printf("Hello world!I have created a dynamic array of 100 integers!\n");

free(p);//Second "bookend" cleans up the space
return 0;
}
```

```
-bash-4.2$ vim hello.c
-bash-4.2$ gcc hello.c -o hello
-bash-4.2$ valgrind ./hello
==113205== Memcheck, a memory error detector
==113205== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==113205== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==113205== Command: ./hello
==113205==
Hello world!I have created a dynamic array of 100 integers!
==113205==
==113205== HEAP SUMMARY:
==113205==     in use at exit: 0 bytes in 0 blocks
==113205==   total heap usage: 1 allocs, 1 frees, 400 bytes allocated
==113205==
==113205== All heap blocks were freed -- no leaks are possible
==113205==
==113205== For lists of detected and suppressed errors, rerun with: -s
==113205== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
-bash-4.2$
```

Now command out the free(p); line in the main.c program and run valgrind again.

```c
#include <stdio.h>
#include <stdlib.h>


int main(int argc, char* argv[])
{
int *p;//Declaration of a pointer variable.

p=(int*)malloc(sizeof(int)*100); //First "bookend" allocates space

printf("Hello world!I have created a dynamic array of 100 integers!\n");

//free(p);//Second "bookend" cleans up the space
return 0;
}
```

```
-bash-4.2$ vim main.c
-bash-4.2$ gcc main.c -o main
-bash-4.2$ valgrind ./main
==115053== Memcheck, a memory error detector
==115053== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==115053== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==115053== Command: ./main
==115053==
Hello world!I have created a dynamic array of 100 integers!
==115053==
==115053== HEAP SUMMARY:
==115053==     in use at exit: 400 bytes in 1 blocks
==115053==   total heap usage: 1 allocs, 0 frees, 400 bytes allocated
==115053==
==115053== LEAK SUMMARY:
==115053==    definitely lost: 400 bytes in 1 blocks
==115053==    indirectly lost: 0 bytes in 0 blocks
==115053==      possibly lost: 0 bytes in 0 blocks
==115053==    still reachable: 0 bytes in 0 blocks
==115053==         suppressed: 0 bytes in 0 blocks
==115053== Rerun with --leak-check=full to see details of leaked memory
==115053==
==115053== For lists of detected and suppressed errors, rerun with: -s
==115053== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
-bash-4.2$ valgrind --leak-check=full ./hello
==115523== Memcheck, a memory error detector
==115523== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==115523== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==115523== Command: ./hello
==115523==
Hello world!I have created a dynamic array of 100 integers!
==115523==
==115523== HEAP SUMMARY:
==115523==     in use at exit: 0 bytes in 0 blocks
==115523==   total heap usage: 1 allocs, 1 frees, 400 bytes allocated
==115523==
==115523== All heap blocks were freed -- no leaks are possible
==115523==
==115523== For lists of detected and suppressed errors, rerun with: -s
==115523== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```