COS 214 Practical 1

Janco Spies, u21434159

July 29, 2022

Task 1

1.1 **a**: Stack, since no dynamic memory has been allocated to the variable.
b: Heap, since the new keyword indicates that dynamic memory was allocated to the variable.
c: Stack, since no dynamic memory has been allocated to the variable.
n: Stack, since no dynamic memory has been allocated to the variable.
d: Stack, since no dynamic memory has been allocated to the variable.
e: Stack, since no dynamic memory has been allocated to the variable.
f: Stack, since no dynamic memory has been allocated to the variable.
g: Stack, since no dynamic memory has been allocated to the variable.
h: Stack, since no dynamic memory has been allocated to the variable.
c[10]: Stack, since no dynamic memory has been allocated to the variable.

1.2 This would not work since NULL is not a valid value for an *int* variable so the value zero will be stored there instead.

1.3 **void* f = (void*) 0xacfe2675;**
This line might not work since whatever value was stored at the memory address "0xacfe2675" cannot necessarily be cast to *void** which might lead to an error.
c[10] = *&*e;
This line might not work since a *char* array is being given the value of an *int* pointer which does not have the same size.
const int* e = (const int*) 522;
This line might not work since e is a pointer pointing to a memory address of a literal, but this literal is not stored there in a variable so following this pointer will lead to a segmentation fault.

Task 2

2.1 The constructor for ClassA is called first for any class derived from ClassA.

2.2 The destructor for ClassA is called last for any class derived from ClassA.

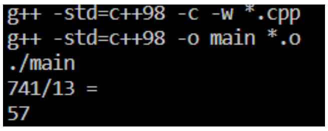
2.3 The constructor of ClassC is called after the constructor of ClassA.

2.4 ClassA then ClassB.

2.5 classB then ClassA.

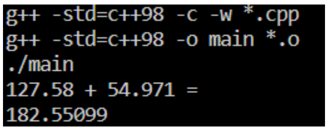
Task 3

3.2



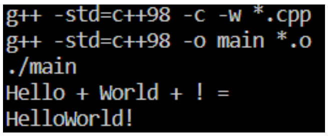
```
g++ -std=c++98 -c -w *.cpp
g++ -std=c++98 -o main *.o
./main
741/13 =
57
```

This worked since the calculator was instantiated with the *int* datatype for which the division operator is defined.

3.3 

```
g++ -std=c++98 -c -w *.cpp
g++ -std=c++98 -o main *.o
./main
127.58 + 54.971 =
182.55099
```

This worked since the calculator was instantiated with the *double* datatype for which the addition operator is defined.

3.4 

```
g++ -std=c++98 -c -w *.cpp
g++ -std=c++98 -o main *.o
./main
Hello + World + ! =
HelloWorld!
```

This worked since the calculator was instantiated with the *string* datatype for which the addition operator is defined.

3.5 This does not work since the multiplication operator is not defined for the *string* datatype.

Task 4

4.1 **cout<<*ptr_a<<”_”<<*ptr_b<<”\n”;**
This line will output ”15_15” since the value ptr_a points to is set to 15 and ptr_b is set to ptr_a, which means that both values point to 15.

4.2 **cout<<*ptr_a<<”_”<<*ptr_b<<”\n”;**
This line will output ”15_4” since ptr_a still points to 15 while ptr_b is set to point to a new value of 4.

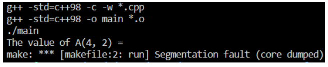
4.3 **cout<<*ptr_a<<”_”<<*ptr_b<<”\n”;**
This line will output ”15_15” since ptr_b’s value that it points to is set to the same value that ptr_a points to, which is 15.

4.4 **cout<<*ptr_a<<”_”<<*&*&*&*&*ptr_b<<”\n”;**
This line will output ”15_15” since after ptr_a is deleted it is set to ptr_b which points to 15. The reference and dereference operators in the cout statement cancel each other out until only the one dereference operator is left.

4.5 **cout<<*ptr_c<<”_”<<**ptr_c<<”\n”;**
This line will output the address of ptr_a followed by ”_15” since ptr_c is set to the address of ptr_a which in turn points to the value 15.

Task 5

5.2 My machine has a limited amount of memory which causes the program to run into a segmentation fault when trying to compute such a large number, even though the implementation works for lesser values of m and n.



```
g++ -std=c++98 -c -w *.cpp
g++ -std=c++98 -o main *.o
./main
The value of A(4, 2) =
make: *** [makefile:2: run] Segmentation fault (core dumped)
```