John Parkhurst

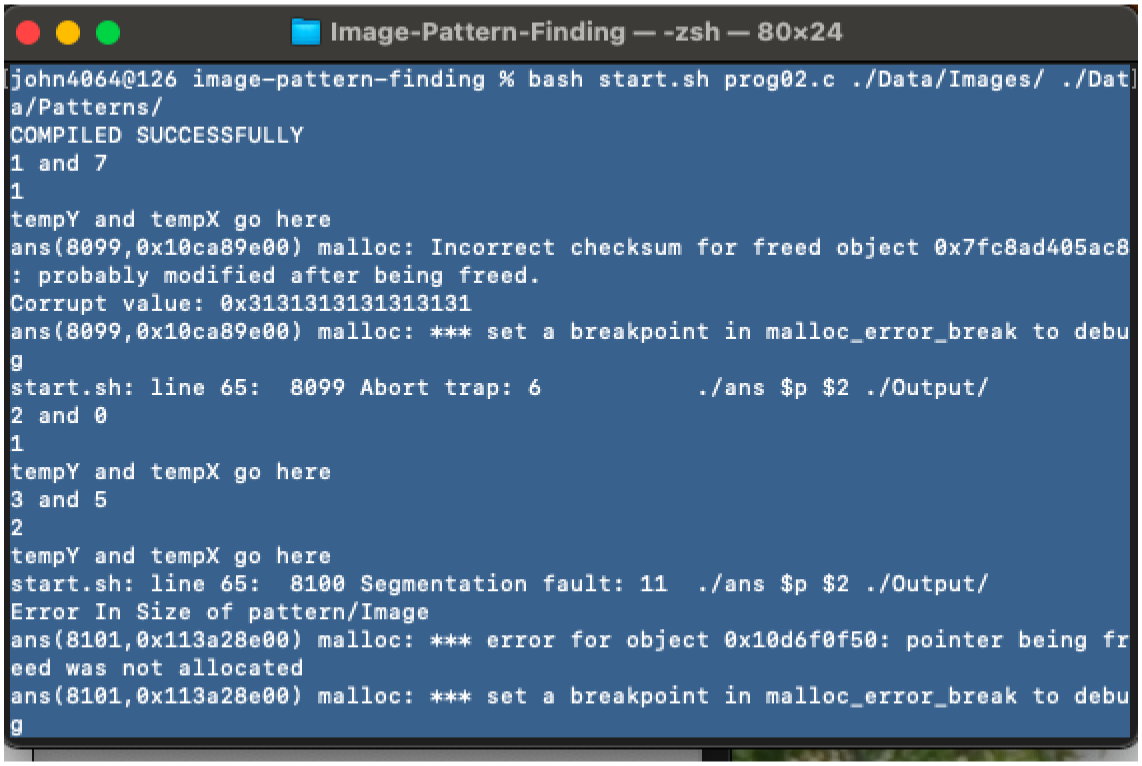
10/13/21

CSC 412

During this assignment, I had to make many decisions in terms of modularity and performance. In regards to modularity, I figured separating the tasks like opening image/pattern files and processing them would be best as separate functions. This allowed me to read the first line of every file which contained their dimensions(calcDimen). Before I processed them so I could properly and adequately allocate them memory. With processing images, I separated that into two distinct functions process and findBlock. The purpose of this was to separate the actions between iterating through each 3x3 combination and the matching process. This; however, is where my two main issues started and I encountered my first serious memory leak. After two days and much debugging I, unfortunately, was not able to fully determine the exact cause of said memory leaks; although, my searching keeps pointing towards blockArr in the process function pointing to ansArr. The program will execute and print out the correct results just at some point it keeps hitting a corrupt value 0x3131313131313131 plus a few other memory management errors. For these reasons I was not able to fully finish the assignment and have it print out the line to the created output files despite being so close.

The performance of this program has some major flaws, to be honest. Passing multiple pointers as parameters for functions such as process and findBloc where they could run thousands of times was a hindsight mistake. It would have been much more beneficial to pass a pointer to a struct that contains the other 3 pointers or necessary data. Given more time to fix these glaring memory errors that would be my next step in refactoring this program. Besides the structs, I think data structures wise it fares fairly well due to the dynamic allocation of space dependent on the size of the image specified on line 1. Now algorithm-wise parsing through the image file for all possible patterns leaves a lot to be desired with my algorithm being O(.5n^2) just for parsing all combinations. The upside is that for every run it'll just grab 9 indexes directly so the actual accessing the elements is O(9). The best performance of my assignment would be the shell script. It performs about as best as possible and as long as the inputs are valid directories/executables it runs with no problem, and if not it will end the application before problems can occur.

Now the limitations of the C program itself include memory issues, the output is limited only to the command prompt, and may not run at all sometimes depending on the runtime. As you can see below many memory issues are limiting the program from fully and properly executing causing it to crash and fail later on. It will display the results of the matches(line 4, line 13, line 16). Given more time I would spend it on fixing the memory issues so it will be able to fully run without any crashes.



```
john4064@126 image-pattern-finding % bash start.sh prog02.c ./Data/Images/ ./Dat
a/Patterns/
COMPILED SUCCESSFULLY
1 and 7
1
tempY and tempX go here
ans(8099,0x10ca89e00) malloc: Incorrect checksum for freed object 0x7fc8ad405ac8
: probably modified after being freed.
Corrupt value: 0x3131313131313131
ans(8099,0x10ca89e00) malloc: *** set a breakpoint in malloc_error_break to debu
g
start.sh: line 65:  8099 Abort trap: 6           ./ans $p $2 ./Output/
2 and 0
1
tempY and tempX go here
3 and 5
2
tempY and tempX go here
start.sh: line 65:  8100 Segmentation fault: 11  ./ans $p $2 ./Output/
Error In Size of pattern/Image
ans(8101,0x113a28e00) malloc: *** error for object 0x10d6f0f50: pointer being fr
eed was not allocated
ans(8101,0x113a28e00) malloc: *** set a breakpoint in malloc_error_break to debu
g
```

Shell script start command: bash start.sh prog02.c ./Data/Images/ ./Data/Patterns/