Name: Fletcher O'Brien
Course: CSCI 312 Principles of Programming Languages
Exam Deadline: May 9, 2025 3:30 PM

Part I - Potpourri

POSIX Definitions

1. Definitions:

   1.1 **Application Program Interface (API)** - The definition of syntax and semantics for providing computer system services.

   1.2 **Built-In Utility** - A utility implemented within a shell. The utilities referred to as special built-ins have special qualities. Unless qualified, the term "built-in" includes the special built-in utilities. Regular built-ins are not required to be actually built into the shell on the implementation, but they do have special command-search qualities.

   1.3 **Executable File** - A regular file acceptable as a new process image file by the equivalent of the *exec* family of functions, and thus usable as one form of a utility. The standard utilities described as compilers can produce executable files, but other unspecified methods of producing executable files may also be provided. The internal format of an executable file is unspecified, but a conforming application cannot assume an executable file is a text file.

   1.4 **Protocol** - A set of semantic and syntactic rules for exchanging information.

   1.5 **Shell** - A program that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal.

   1.6 **Standard Error** - An output stream usually intended to be used for diagnostic messages.

   1.7 **Standard Input** - An input stream usually intended to be used for primary data input.

   1.8 **Standard Output** - An output stream usually intended to be used for primary data output.

   1.9 **Terminal** - A character special file that obeys the specifications of the general terminal interface.

   1.10 **Utility** - A program, excluding special built-in utilities provided as part of the Shell Command Language, that can be called by name from a shell to perform a specific task, or related set of tasks.

Part II - Bash

Learning the Shell

2. -

3. Types of commands:

Type	Example
alias	alias cls='clear'
shell builtin	echo
shell function	example(){ echo "Hello" }
executable	example.exe

4. Redirection operators:

Redirection	Operator
Redirecting input	<
Redirecting output	>
Redirecting error	2>
Appending redirected output	>>
Redirecting standard output and standard error	&>
Appending standard output and standard error	&¿¿

5. Types of expansion:

   1.1 Pathname Expansion

   1.2 Tilde Expansion

   1.3 Arithmetic Expansion

   1.4 Brace Expansion

   1.5 Parameter Expansion

   1.6 Command Substitution

6. Types of quoting:

Type	What does it suppress?
Double Quotes	Special Characters dollar sign, backslash, and backtick/accentgrave
Single Quotes	All expressions
Escaping Characters	Everything except a single character

7. Types of files:

Name	Attribute	Type
file1		empty file
file2	.o	Pathname
file3	.h	Tilde Expansion
file4	.out	Arithmetic Expansion
file5	TODO	TODO
file6	TODO	TODO
file7	TODO	TODO

8. Type of file, ie - for regular file, d for directory, l for symlink
9. Owner can read, write, and execute. Others can only read.
10. Owner has full access, others can only execute.
11. Owner has full access, others can only read and execute.
12. Owner can only read and write, not execute. Others have no access.
13. Everyone can read, write, and execute.
14. I have read permission on the directory but not the file.
15. I have execute permissions on the directory but not read permissions.
16. Write doesn't work because I don't have owner on other's terminals, but wall works because it is owned by a group that does have that permission.

Configuration and the Environment

17. Types of data:
    1.1 Function
    1.2 Variable
18. When you type a command that doesn't have a slash the shell searches directories in order.
19. LD_LIBRARY_PATH
20. Types of sessions:
    1.1 Login
    1.2 Non-Login
21. Startup files:
    1.1 bash files
    1.2 profile files

Common Tasks and Essential Tools

22. Basic treats special characters as ordinary unless you use backslash, ERE treats them all as special unless you backslash.

Part III - C

Maintain your perceptron ADT

23. Valgrind runs your program on a synthetic CPU and helps programmers check for memory leaks.

24.
```
    =3075125== Memcheck, a memory error detector
==3075125== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Sewar
==3075125== Using Valgrind-3.19.0 and LibVEX; rerun with -h for cop
==3075125== Command: ./a.out train.dat
==3075125==
==3075125== Invalid read of size 1
==3075125==    at 0x48BBD83: strtodlinternal (in /usr/lib64/libc.so
==3075125==    by 0x4013F4: atof (stdlib-float.h:27)
==3075125==    by 0x4013F4: newData (perceptron.c:59)
==3075125==    by 0x401209: main (main.c:15)
==3075125==  Address 0x0 is not stack'd, malloc'd or (recently) fre
==3075125==
==3075125==
==3075125== Process terminating with default action of signal 11 (S
==3075125==  Access not within mapped region at address 0x0
==3075125==    at 0x48BBD83: /strtodlinternal (in /usr/lib64/libc.s
==3075125==    by 0x4013F4: atof (stdlib-float.h:27)
==3075125==    by 0x4013F4: newData (perceptron.c:59)
==3075125==    by 0x401209: main (main.c:15)
==3075125==  If you believe this happened as a result of a stack
==3075125==  overflow in your program's main thread (unlikely but
==3075125==  possible), you can try to increase the size of the
==3075125==  main thread stack using the --main-stacksize= flag.
==3075125==  The main thread stack size used in this run was 838860
==3075125==
==3075125== HEAP SUMMARY:
==3075125==     in use at exit: 5,384 bytes in 6 blocks
==3075125==   total heap usage: 6 allocs, 0 frees, 5,384 bytes allo
==3075125==
==3075125== 24 bytes in 1 blocks are still reachable in loss record
==3075125==    at 0x484486F: malloc (vgreplacemalloc.c:381)
==3075125==    by 0x401324: newData (perceptron.c:31)
==3075125==    by 0x401209: main (main.c:15)
```

```
==3075125==
==3075125== 160 bytes in 1 blocks are still reachable in loss recor
==3075125==    at 0x484486F: malloc (vgreplacemalloc.c:381)
==3075125==    by 0x4013B5: newData (perceptron.c:47)
==3075125==    by 0x401209: main (main.c:15)
==3075125==
==3075125== 160 bytes in 1 blocks are still reachable in loss recor
==3075125==    at 0x484486F: malloc (vgreplacemalloc.c:381)
==3075125==    by 0x4013C1: newData (perceptron.c:48)
==3075125==    by 0x401209: main (main.c:15)
==3075125==
==3075125== 472 bytes in 1 blocks are still reachable in loss recor
==3075125==    at 0x484486F: malloc (vgreplacemalloc.c:381)
==3075125==    by 0x48EB04E: fopeninternal (in /usr/lib64/libc.so.6
==3075125==    by 0x401312: newData (perceptron.c:25)
==3075125==    by 0x401209: main (main.c:15)
==3075125==
==3075125== 472 bytes in 1 blocks are still reachable in loss recor
==3075125==    at 0x484486F: malloc (vgreplacemalloc.c:381)
==3075125==    by 0x40145E: newData (perceptron.c:55)
==3075125==    by 0x401209: main (main.c:15)
==3075125==
==3075125== 4,096 bytes in 1 blocks are still reachable in loss rec
==3075125==    at 0x484486F: malloc (vgreplacemalloc.c:381)
==3075125==    by 0x48EA723: IOfiledoallocate (in /usr/lib64/libc.s
==3075125==    by 0x48F83BF: IOdoallocbuf (in /usr/lib64/libc.so.6)
==3075125==    by 0x48F74EB: IOfileunderflow@@GLIBC2.2.5 (in /usr/l
==3075125==    by 0x48F8475: IOdefaultuflow (in /usr/lib64/libc.so.
==3075125==    by 0x48EBE2B: IOgetlineinfo (in /usr/lib64/libc.so.6
==3075125==    by 0x48EAF0F: fgets (in /usr/lib64/libc.so.6)
==3075125==    by 0x40137D: newData (perceptron.c:36)
==3075125==    by 0x401209: main (main.c:15)
==3075125==
==3075125== LEAK SUMMARY:
==3075125==    definitely lost: 0 bytes in 0 blocks
==3075125==    indirectly lost: 0 bytes in 0 blocks
==3075125==      possibly lost: 0 bytes in 0 blocks
==3075125==    still reachable: 5,384 bytes in 6 blocks
==3075125==         suppressed: 0 bytes in 0 blocks
==3075125==
==3075125== For lists of detected and suppressed errors, rerun with
==3075125== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0
Segmentation fault (core dumped)
```

25. -

26. -

Evolve your perceptron ADT

27. 90%

Part IV - Python

Evolve a simple interpreter

28. -

Part V - Java

Describe the functionality of a class

29. Im doing 1