

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page. The exam has 10 questions with equal weight. Some questions have multiple parts. Point totals are given in the margin.

Name: \_\_\_\_\_ **Answer Key** \_\_\_\_\_

1. \_\_\_\_\_ **10**
  2. \_\_\_\_\_ **10**
  3. \_\_\_\_\_ **10**
  4. \_\_\_\_\_ **10**
  5. \_\_\_\_\_ **10**
  6. \_\_\_\_\_ **10**
  7. \_\_\_\_\_ **10**
  8. \_\_\_\_\_ **10**
  9. \_\_\_\_\_ **10**
  10. \_\_\_\_\_ **10**
- Total \_\_\_\_\_ **100**

No notes or books may be used on the exam. If you have any questions, please raise your hand and I will try to answer them.

1. Indicate whether each of the following statements is True (T) or False (F); for example:

(a) This class was:

- i.   T   fun and exciting.
- ii.   F   taught by a complete idiot.
- iii.   F   a complete waste of time.
- iv.   T   taught by someone with a strange sense of humor.

2

(a) Blackboard architectures:

- i.   F   allow direct communication between clients.
- ii.   T   are exemplified by communicating applets.
- iii.   T   are easily extended.
- iv.   T   gracefully tolerate the failure of a client.

2

(b) Event-driven architectures:

- i.   F   are easy to debug.
- ii.   F   are difficult to extend.
- iii.   T   are exemplified by windowing systems.
- iv.   F   have strong coupling between components.

2

(c) Layered architectures:

- i.   T   may have serious performance problems.
- ii.   T   can be used to model network protocol stacks.
- iii.   F   support abstraction poorly.
- iv.   T   are easily extended.

2

(d) Object-oriented architectures:

- i.   T   support encapsulation.
- ii.   T   have a high degree of reuse.
- iii.   F   are exemplified by the UNIX operating system.
- iv.   F   do not encourage extendibility.

2

(e) Pipe and filter architectures:

- i.   T   exhibit low coupling.
- ii.   F   are very efficient.
- iii.   F   are well-suited to interactive applications.
- iv.   F   allow sharing of global information between components.

2. Describe what each of the following UNIX utilities does; for example:

(a) cat

**Answer:** Copies the files given as arguments, in order, to standard output. If no arguments are given, standard input is copied to standard output.

2

(a) basename

**Answer:** Removes any leading directory components from the filename given as the first argument, and writes the result to standard output. Additionally, a suffix may be specified as the second argument, and if the suffix is identical to the end of the filename, it is removed.

2

(b) dirname

**Answer:** Writes to standard output the directory component of the filename given as the first argument. If the filename has no directory component, then a single period is written.

2

(c) grep

**Answer:** Searches for lines matching a pattern specified as the first argument in the files specified as the remaining arguments. If no files are given, the standard input is searched. Matching lines are written to standard output.

2

(d) head

**Answer:** Copies to standard output the first part of each file given as an argument. If no arguments are given, the standard input is used. By default, the first ten lines are copied. The default number of lines may be changed using an option given on the command line.

2

(e) tail

**Answer:** Copies to standard output the last part of each file given as an argument. If no arguments are given, the standard input is used. By default, the last ten lines are copied. The default number of lines may be changed using an option given on the command line.

3. Define each of the following terms:

2

(a) extendible

**Answer:** capable of being built upon

2

(b) extensible

**Answer:** capable of being used for purposes other than the intended

2

(c) maintainable

**Answer:** capable of being easily changed or evolved over time

2

(d) module

**Answer:** a physical or logical decomposition of code input a set of inputs, outputs, and behaviors

2

(e) portable

**Answer:** capable to running with minor or no modifications on different platforms

4. Write a SED script that performs the following actions:

2

(a) Deletes all blank lines from its input.

**Answer:**

```
/^$/d
```

2

(b) Removes any uppercase letter from each input line.

**Answer:**

```
s/[A-Z]//g
```

2

(c) Places the string <BR> after each blank line.

**Answer:**

```
/^$/a\  
<BR>
```

2

(d) Ensures that exactly two spaces follow a period if it is preceded by a lowercase letter.

**Answer:**

```
s/\([a-z]\)\. * /\1.  /g
```

2

(e) Creates a file called `output` containing the first ten lines of the input.

**Answer:**

```
1,10w output
```

5. Write an AWK script or a shell script using `awk` and `sort` to perform the following actions:

USER	PID	%CPU	%MEM	SZ	RSS	TT	S	START	TIME	COMMAND
root	23353	0.4	0.1	2144	1904	?	S	17:36:01	0:03	[ sshd1 ]
root	23436	0.1	0.1	1560	1152	pts/6	O	17:36:49	0:00	/usr/ucb/ps aux
atkinson	23357	0.1	0.1	2144	1744	pts/6	S	17:36:06	0:00	-tcsh
root	257	0.0	0.1	4440	4176	?	S	May 17	2:54	/usr/sbin/nscd
jfan	22778	0.0	0.1	2272	1800	pts/3	S	17:11:58	0:00	-bash
daemon	308	0.0	0.1	10376	3264	?	S	May 17	0:00	/usr/lib/ab2/dweb/
web	10139	0.0	0.8	45608	29256	?	S	13:09:20	0:12	/opt/jdk-1.3.0/sol
dfabris	15000	0.0	0.1	2272	1800	pts/2	S	May 22	0:00	-bash
nobody	15224	0.0	0.1	3200	2072	?	S	May 17	0:00	fs
oracle8i	18034	0.0	0.2	91064	7912	?	S	Jun 01	0:00	ora_d000_DC81

2 (a) Write the starting date or time of all processes.

**Answer:**

```
NR > 1 {if ($9 ~ /^[0-9]/) print $9; else print $9, $10}
```

2 (b) Write the PIDs of all processes owned by root in sorted order.

**Answer:**

```
#!/bin/sh
awk '$1 == "root" {print $2}' | sort -n
```

2 (c) Write the total size (SZ) used by processes owned by root.

**Answer:**

```
NR > 1 && $1 == "root" {sum += $5}
END {print sum}
```

2 (d) Write the total number of processes owned by each user.

**Answer:**

```
NR > 1 {count [$1] ++}
END {for (user in count) print user, count [user]}
```

2 (e) Write the data sorted by username.

**Answer:**

```
#!/bin/sh
awk 'NR > 1' | sort
```

6. Answer the following questions regarding RCS and DIFF:

2

(a) Indicate whether each of the following RCS version numbers is legal:

i.   Y   1.2

ii.   N   1.0

iii.   Y   1.3.4.1

iv.   N   1.2.1

2

(b) How many read-only copies of an RCS-controlled file can exist at any one time?

**Answer:** unlimited

2

(c) How many write-only copies of an RCS-controlled file can exist at any one time?

**Answer:** one

2

(d) The command “diff a b” produces output for constructing file b from file a. How would you produce output for constructing file a from file b?

**Answer:**

```
diff b a
```

2

(e) Use DIFF to check if two files are identical. If the files are identical, write the single word okay to standard output. Otherwise, write nothing.

**Answer:**

```
diff file1 file2 >/dev/null 2>&1 && echo okay
```

7. Answer the following questions related to web-based application development:

2

(a) Why should the server application check the validity of a form even if JavaScript is used?

**Answer:** The client may have JavaScript disabled or the JavaScript code may not be fully compatible with the client's implementation.

2

(b) What is meant when we say that HTTP is a stateless protocol?

**Answer:** The protocol does not provide any mechanism for remembering information or state between connections. Each request must include all necessary information.

2

(c) Why is security a big consideration when designing web-based tools?

**Answer:** A client may be required to download unknown code to the client machine; a server may be required to accept connections containing arbitrary data from clients. Both situations allow the possibility of an unknown agent gaining access to a machine.

2

(d) What is the format of an HTTP response such as that sent by a CGI script?

**Answer:** The response consists of an optional content header, followed by a blank line, followed by the actual content.

2

(e) List three different content types.

**Answer:**

- text/html
- text/plain
- image/gif

8. Answer the following questions regarding servlets:

2

(a) Why does servlet code need to be multi-thread safe?

**Answer:** A servlet needs to be multi-thread safe because each connection to the servlet creates a new thread that runs through a single instance of the class.

2

(b) How does a servlet retrieve the parameters of a form?

**Answer:** A servlet calls `getParameter()` or `getParameterValues()`.

2

(c) How do class and instance variables relate to the number of connections to the servlet?

**Answer:** All connections to a servlet via the same path share instance variables. All connections to a servlet regardless of path share class variables.

2

(d) What is a cookie?

**Answer:** A cookie is a name and value pair used to establish a session.

2

(e) List three methods provided by the response object.

**Answer:**

- `addCookie()`
- `getWriter()`
- `setContentType()`

- 5 9. (a) Write a shell script called `ans` that reads a line from standard input and exits with a zero status if the first character on the line starts with a `y` or `Y`, and exits with a nonzero status otherwise.

**Answer:**

```
#!/bin/sh

read LINE
echo $LINE | grep "^[yY]" >/dev/null 2>&1
```

- 5 (b) Write a shell script called `seq` that prints a sequence of integers to standard output. The starting and ending values are given as arguments. The starting value may be omitted, in which case it defaults to one. For example:

```
$ seq 3
1
2
3
$ seq 5 8
5
6
7
8
```

**Answer:**

```
#!/bin/sh

if [ $# -eq 1 ]; then
    START=1
    END=$1
else
    START=$1
    END=$2
fi

while [ $START -le $END ]; do
    echo $START
    START=`expr $START + 1`
done
```

- 5 10. (a) Consider developing a web-based chat system that allows multiple users to talk with one another. After connecting to the server, a user would enter text in one area and that text would be seen in another area on all clients. What technologies would you use for the client? What technologies would you use for the server? Briefly justify your answer.

**Answer:** Applets on the client would be the best choice since the client needs to have dynamic, rather than static, content. Servlets on the server would be the best choice since the server application needs to remember the list of applets with which it should communicate. It would also be inefficient to connect to and run a CGI application repeatedly to send or retrieve lines of text. If HTTP is used, either GET or POST may be used. Alternatively, a raw socket connection could be established to the servlet by each applet.

- 5 (b) Consider developing a web-based application for allowing a user to check his or her Design Center e-mail remotely using any web browser as a client. What technologies would you use for the client? What technologies would you use for the server? Briefly justify your answer.

**Answer:** The client would consist of HTML forms, possibly supplemented by JavaScript. This approach is more portable than applets and is a better choice since the client must work correctly on any web browser. The server technologies would consist of several CGI applications rather than servlets. Although servlets easily support session tracking, the security considerations of servlets preclude their use in this situation. A CGI application could easily be run using the correct Design Center userid. Finally, the POST method, preferably over HTTPS, should be used, since the request may have sensitive information.