Question 1: Bash Expressions

Provide a syntactically correct example of a bash statement that satisfies the question. Write your answer within the box provided. Do not write your answer in the exam booklet. Your answer must fit within the provided space below.

Problem	Solution
Copy all the files that have a 3-letter extension where the last letter is C. Copy these files to the subdirectory Sol1 that is below the current directory.	
Concatenate all the text files in the current directory that have at least one occurrence of the word BOB (in any case) within the text of the file.	
If the information in $1 is a file name and the information in $1 contains the characters DOC in upper case, then chmod the file to permit it to be editable by those in your group.	
Count the number of times the word BOB (in any case) occurs in a text file.	

Question 2: GNU Problems

Question 2.1: Makefiles

Assume there is a software project with three programmers: P1, P2, and P3. P1 is the team leader and is interested in constructing a project that facilitates development of a complex application written in C. P1 decides that this should be a modular program and that a makefile should be used.

(A) Write a makefile that satisfies these concerns:

- P1 programs: main.c and database.c. P1 needs access to menu.o
- P2 programs: menu.c and divide+conquer.c. P2 needs access to all of P3's programs and P1's database.o from divide+conquer.c
- P3 programs: fact.c, fib.c, csv.c, and library.c
- Minimize the number of source files needed (both .c and .h)
- Compile should be as quick as possible
- There needs to be an optional feature to backup files into a directory called mybackup, creating that directory if it does not already exist in the current directory
- There needs to be an optional delete all the .o files feature

(B) P1 wants to make sure that no one includes the same header file twice. Give an example of how to write a header file so that the file cannot be loaded twice during compile.

Question 2.2: Profiling

PART 1: Given this profiling output:

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 80.64    183.25   183.25      500     0.37     0.38  match
 14.79    216.86    33.61      554     0.06     0.07  train_match
  1.93    221.25     4.39     4922     0.00     0.00  simtest2
  1.32    224.26     3.01     4922     0.00     0.00  reset_nodes2
  0.47    225.33     1.07      554     0.00     0.00  weightadj
  0.32    226.05     0.72     1054     0.00     0.00  reset_nodes
  0.32    226.77     0.72      554     0.00     0.00  simtest
  0.11    227.02     0.25 11080000     0.00     0.00  g
  0.07    227.17     0.15        1     0.15   191.15  scan_recognize
  0.01    227.19     0.02        3     0.01     0.01  init_bu
```

Answer the following questions:

a) Identify the two fastest functions and why.
b) Identify the two slowest functions and why.

PART2: Reconstruct the C program functions and how they are called given the Profiler Call Graph below and identify the slowest function (justify):

```
index % time    self  children    called     name
                                               <spontaneous>
[1]    100.0    0.00    0.92                 main [1]
                0.06    0.85       1/1           test [2]
                0.01    0.00       1/1           some_other_test [5]
-----------------------------------------------
                0.06    0.85       1/1           main [1]
[2]     98.9    0.06    0.85       1         test [2]
                0.00    0.85       1/1           another_test [3]
-----------------------------------------------
                0.00    0.85       1/1           test [2]
[3]     92.3    0.00    0.85       1         another_test [3]
                0.85    0.00       1/1           yet_another_test [4]
-----------------------------------------------
                0.85    0.00       1/1           another_test [3]
[4]     92.3    0.85    0.00       1         yet_another_test [4]
-----------------------------------------------
                0.01    0.00       1/1           main [1]
[5]      1.1    0.01    0.00       1         some_other_test [5]
-----------------------------------------------
```

In other words, write the program in C. Write the main() function, the test() function, the some_other-test() function, etc for all the above functions. From within each function write the appropriate calls to the functions it references. You cannot provide any other code since that information is not provided.

Question 2.3: GDB

Assume we have this functions: int strlen(char *ptr);
The above function returns the length of a null terminating string. Assume a user did not provide a null terminating string. Assume the user does not realize they did not provide a null terminating string.

Write down all the commands from an imaginary GDB session showing how the programmer could use GDB to discover their error. (1) Show how they would compile their code to be compatible with GDB. (2) Show how they would launch GDB with their program. (3) Show a portion of the GDB session that locates and diagnoses the problem, demonstrating why this problem exists.

Question 2.4: Git

Assume question 2.1 from above. P1 now wants to arrange a code repository for the project. Each programmer has their own laptop. The project has a server where the application will be deployed and the project source code will be stored.

P1 wants a Git solution that handles the following problems, describe how Git can help by providing the Git commands you would use:

a) Create the shared and local Git repository
b) Determine rules/procedures for using the repository (push, pull and update rules), P1 wishes to impose these rules on the other programmers.
c) Developing code in versions. Using a branch to update an existing version, then merging that branch into the existing version, and updating the version number.
d) Suggest two other things P1 should do with Git to improve the team development experience under Git.

Question 3: C Programming Problem

A programmer wants a data structure to store all the people who belong to a website in RAM. There are two types of users: members and system operators. Members have a user name, password, and group name; all three are strings of size 30 or less. System operators have a user name, password, and a security level (integer).

Answer the following questions using C.

a) Write an optimal data structure that can store a single user, based on the above description.
b) Assuming we can calculate the total number of users, write the C code that would build at run-time a data structure that could be used to store all those users (using your data structure from (a)).
c) Assuming all the users are stored in a CSV file with the following format:
   type, user name, password, parameter
   Where type = M or S for member or system operator; user name and password are obvious, and parameter = group name if the type is M, or security level if the type is S.
   Write C code that would read from the CSV and populate your data structure from (b) with data existing in the CSV file (assume the file is named user.csv).
d) Write some code that will fork() a program into two threads to search the data structure (from (b)) in RAM twice as fast. To make it easy the search can be a simple incremental for-loop (a linear search).