Name: ___________________________________  Student Number:__________________________

READ THESE INSTRUCTIONS

This is an open-book exam. The use of the notes contained in your class portfolio is permitted. The questions on this examination require you to provide answers similar to those you have seen in lecture, used in the course homework assignments, or presented in the course textbook.

Answer all questions in the spaces indicated and only in the spaces indicated. Read each question carefully and work as rapidly as you can without being careless.

Only questions about these instructions, typing, or copying errors will be answered during the exam. If you are not sure what a question means, state your interpretation and proceed. Use the most direct interpretation and select the best answer for that interpretation.

Academic Honesty

At the end of the examination, please sign:

In recognition of and in the spirit of the Wright State University policies of academic honesty, I certify that I have neither given nor received unpermitted aid in this examination.

Signature:_____________________________

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO
1. Of the phases a C program has to pass through in order to become a standalone executable, which of the phases listed below happens first?
   a. assembling
   b. compiling
   c. linking
   d. loading
   e. sourcing

2. Which of the following phases produces, as its output, an object file (*.o)?
   a. assembling
   b. compiling
   c. linking
   d. loading
   e. pre-processing

3. What is the purpose of the assembly phase?
   a. transform assembled code to object code
   b. transform pre-processed code into assembly code
   c. produce an executable file
   d. append necessary object files/libraries
   e. none of the above

4. Which of the following statements regarding object files is false? Object files:
   a. can be recreated without modifying other object files
   b. can be executed directly
   c. can have their modifications dates compared to corresponding program files to determine if recompilation is necessary
   d. can be combined to form a static library or archive
   e. can be combined to form a dynamic/shared library or archive

5. Which of the following tools would automatically flag potential problems such as using values returned from a function call without first testing the value for an error condition?
   a. gcc
   b. lint
   c. truss
   d. make
   e. gdb

6. Which of the following tools would automatically analyzes syntax errors while largely ignoring semantic errors?
   a. gcc
   b. lint
   c. truss
   d. make
   e. gdb

7. Which of the following tools during normal operation will automatically tell you which system calls are executed by your program?
   a. gcc
   b. lint
   c. truss
   d. make
   e. gdb

8. Which flag must you use when compiling your code to ensure that gdb can be used effectively?
   a. -c
   b. -d
   c. -D
   d. -g
   e. -x

9. Which of the following symbols could be used (in tcsh) to pipe both stderr and stdout through the same filter?
   a. |<
   b. &!
   c. ||
   d. &&
Use the typescript listed below when answering the questions on this page.

<unix> more datafile1
ee:451:Digital System Design
eee:651:Digital System Design
ceg:333:Introduction to Unix
ceg:320:Computer Organization & Assembly Language Programming
ceg:520:Computer Organization & Assembly Language Programming
ceg:360:Digital System Design
ceg:560:Digital System Design
<unix> command1
<unix> more datafile2
eee:451:Digital System Design
eee:651:Digital System Design
ceg:333:Introduction to Unix
ceg:320:Computer Organization & Assembly Language Programming
ceg:520:Computer Organization & Assembly Language Programming
ceg:360:Digital System Design
ceg:560:Digital System Design
<unix> command2
<unix> more datafile3
ceg:320:Computer Organization & Assembly Language Programming
ceg:333:Introduction to Unix
ceg:360:Digital System Design
ee:451:Digital System Design
ceg:520:Computer Organization & Assembly Language Programming
ceg:560:Digital System Design
ee:651:Digital System Design
<unix> command3
<unix> more datafile4
Computer Organization & Assembly Language Programming
Digital System Design
Introduction to Unix

10. Which of the following commands, when executed in place of command1 above, will result in the datafile2 shown above?
a. grep ' -: ' < datafile1 > datafile2
b. grep -v ' -: ' datafile1 > datafile2
c. tr -s ' -: ' datafile1 > datafile2
d. tr ' -: ' ' -: ' < datafile1 > datafile2
e. tr -s ' -: ' ' -: ' < datafile1 > datafile2

11. Which of the following commands, when executed in place of command2 above, will result in the datafile3 shown above?
a. cut -t -k1,1 datafile2 > datafile3
b. grep -t: -n -k1,1 datafile2 > datafile3
c. tr -t: -k2,2 datafile2 > datafile3
d. sort -t: -n -k2,2 datafile2 > datafile3
e. wc -t: -k1,2 datafile2 > datafile3

12. Which of the following sequences of commands, when executed in place of command3 above with appropriate arguments, could result in the datafile4 shown above?
a. wc (args) | uniq (args)
b. sort (args) | grep (args) | cut (args)
c. cut (args) | sort (args) | uniq (args)
d. grep (args) | wc (args) | tr (args)
e. sort (args) | uniq (args) | grep (args)
Use the directory listing and makefile listed below when answering the questions on this page.

<unix> ls -l
-rw------- 1 w001ted users 158 Oct 22 21:55 Makefile
-rw------- 1 w001ted users 2306 Oct 22 21:54 doomC.c
-rw------- 1 w001ted users 19 Oct 22 21:57 doomC.h
-rw------- 1 w001ted users 1236 Oct 22 21:57 doomC.o
-rwx------- 1 w001ted users 7216 Oct 22 21:57 lab01
-rw------- 1 w001ted users 2309 Oct 22 21:56 lab01.c
-rw------- 1 w001ted users 25 Oct 22 21:58 lab01.h
-rw------- 1 w001ted users 1264 Oct 22 21:57 lab01.o

<unix> more Makefile
lab01: doomC.o lab01.o
gcc -o lab01 doomC.o lab01.o
lab01.o: lab01.h lab01.c
gcc -c lab01.o
doomC.o: doomC.h doomC.c
gcc -c doomC.c
clean:
    rm -f core *.o

13. What is the default target for the listed makefile?
   a. lab01
   b. lab01.o
   c. doomC.o
   d. clean
   e. $(CC) –o $@ $(CFLAGS) $(OBJECTS)

14. The creation of what file is the ultimate target of this makefile?
   a. lab01
   b. lab01.o
   c. doomC.o
   d. clean
   e. $(CC) –o $@ $(CFLAGS) $(OBJECTS)

15. Given the Makefile and directory listing above (note modification times), how many separate invocations to gcc will be made if you type "make" on the command line?
   a. 0
   b. 1
   c. 2
   d. 3
   e. more than 3

16. Which of the files listed below would be removed if you typed "make clean" on the command line?
   a. lab01
   b. lab01.o
   c. lab01.h
   d. doomC.h
   e. all of the above
Consider the following segment of a gdb session when answering the questions on this page:

Program received signal SIGSEGV, Segmentation fault.
0xff2b31bc in strlen () from /usr/lib/libc.so.1
(gdb) bt
#0 0xff2b31bc in strlen () from /usr/lib/libc.so.1
#1 0xff3061c8 in _doprnt () from /usr/lib/libc.so.1
#2 0xff307c70 in fprintf () from /usr/lib/libc.so.1
#3 0x000109c0 in printList (list=0xffbef374, n=3) at bomber.c:66
#4 0x00010918 in main (argc=3, argv=0xffbef374, environ=0xffbef384)
  at bomber.c:38
(gdb) up 4
#4 0x00010918 in main (argc=3, argv=0xffbef374)
  at bomber.c:38
38    printList(argv,argc);

17. What is the execution status of this program?
   a. This program is running normally.
   b. This program is running, but is in an infinite loop.
   c. This program has crashed due to a syntax error.
   d. This program has crashed due to an illegal memory reference.
   e. This program has crashed due to over-recursion.

18. What is the name of the lowest-level function most recently called for execution?
   a. _doprnt ()
   b. fprintf()
   c. main()
   d. printList()
   e. strlen()

19. What is the name of the lowest-level function that exists in the source file created by the pro-
    grammer/user that was most recently called for execution?
   a. _doprnt ()
   b. fprintf()
   c. main()
   d. printList()
   e. strlen()

20. How many arguments were passed to the function main?
   a. 1
   b. 2
   c. 3
   d. 4
   e. more than 4
21-30. [10 points] Write a makefile to create the executable file exam02 with the following requirements and dependencies. Note: feel free to use .CC files and the g++ compiler instead of .c files and the gcc compiler if you feel more comfortable using C++ conventions for this example.

**Requirements**

- exam02 is built from exam02-1.o, exam02-2.o, mods.o
- mods.o does not need to be created
- exam02-1.c includes exam02.h
- exam02-2.c includes exam02.h
- use gcc/g++ compiler with gdb flag
- set shell environment variable to 'tcsh'
- include a target (clean) to "clean" the directory of temporary files created during the make process
- include a target (all) to compile everything

**Points Breakdown**

- (3 points) create exam02, exam02-2.o, exam02-2.o targets
- (1 point) create "clean" target
- (1 point) create "all" target
- (2 points) correct use of compiler and flag(s)
- (1 point) set shell environment variable
- (2 points) descriptive comments for each line