

Name _____

Please circle your section number:

018 (12:20pm lab)

019 (1:25pm lab)

020 (2:30pm lab)

021 (3:35pm lab)

General Instructions

- DO NOT WRITE YOUR NAME ON ANY PAGE EXCEPT THIS ONE!
- You have two hours. **Pace yourself**, and pay attention to the point values.
- The exam is 67% multiple choice, and 33% programming and short answer.
The programming/short answer questions start with number 27.
- Read *all* the directions *carefully* on each problem.
- Good luck.

Multiple Choice: Mark answers on Scantron in #2 pencil

Project 3 error messages

Instructions: A function definition for `chooseTableWithFavoriteServer` appears on the following page. Note that at three points in the code there are calls to the `printf` function that include the words `**** ERROR MESSAGE ****` and then a number: #1, #2, or #3.

Match each occurrence of `**** ERROR MESSAGE ****` with a more suitable error text.

1. (3 pts) `**** ERROR MESSAGE **** #1`

- (a) There are no free tables in the restaurant
- (b) Your favorite server has no free tables with enough space
- (c) Your favorite server was not found in the list of servers
- (d) Your favorite server has not been assigned any tables
- (e) Number of guests cannot be a negative number

2. (3 pts) `**** ERROR MESSAGE **** #2`

- (a) There are no free tables in the restaurant
- (b) Your favorite server has no free tables with enough space
- (c) Your favorite server was not found in the list of servers
- (d) Your favorite server has not been assigned any tables
- (e) Number of guests cannot be a negative number

3. (3 pts) `**** ERROR MESSAGE **** #3`

- (a) There are no free tables in the restaurant
- (b) Your favorite server has no free tables with enough space
- (c) Your favorite server was not found in the list of servers
- (d) Your favorite server has not been assigned any tables
- (e) Number of guests cannot be a negative number

```

void chooseTableWithFavoriteServer(int table[],
                                   int tableCapacities[],
                                   int tableToServer[], int numServers,
                                   int numTables,
                                   char serverName[][SERVER_NAME_MAX])
{
    int i, j, guests;
    char name[SERVER_NAME_MAX];

    printf("Please enter the name of your favorite server: ");
    fgets(name, SERVER_NAME_MAX, stdin);

    printf("How many guests? (0 to cancel): ") ;
    guests = readFirstInteger();
    while (guests < 0)
    {
        printf("Sorry: **** ERROR MESSAGE **** #1\n");
        printf("How many guests? (0 to cancel): ");
        guests = readFirstInteger();
    }
    if (guests==0)
        return;
    for (j=0; j<numServers; j++)
    {
        if (0 == strcmp(name, serverName[j]))
        {
            for (i=0; i<numTables; i++)
            {
                if (tableToServer[i]==j && table[i]==0
                    && tableCapacities[i]>=guests)
                    break;
            }

            if (i<numTables)
            {
                table[i] = guests;
                printf("There are now %d guests seated at table %d.\n", guests, i);
                return;
            }
            printf("Sorry: **** ERROR MESSAGE **** #2\n");
            return;
        }
    }
    printf("Sorry: **** ERROR MESSAGE **** #3\n");
    return;
}

```

Command Line arguments

Instructions: Suppose you are writing a program to convert Celsius to Farenheit. Rather than prompt the user for the celsius temperature, you want to accept the celsius temperature as a command line argument. For example, this might be a sample script (Assume that `strauss>` is the Unix command prompt.)

```
strauss> ./convert 0
32 degrees Farenheit
strauss>
```

If you pass in no command line arguments, you'll give the user a message:

```
strauss> ./convert
Usage: ./convert celsiusTemp
strauss>
```

If you pass in too many command line arguments, you'll give the user the same message:

```
strauss> ./convert 12 foo bar fum
Usage: ./convert celsiusTemp
strauss>
```

When the wrong number of command line arguments is passed, the following code could be used to print the error message. However, some parts of this code have been replaced with two nonsense symbols: `@@@@@` and `^^^^^`. (The questions which follow ask you to change these nonsense symbols to correct C code.)

```
if (@@@@@)
{
    printf("Usage: %s celsiusTemp\n",^^^^^);
    exit(-1);
}
```

Now, answer the following questions about this program (see next page).

4. (2 pts) What should be put in place of @@@@?

- (a) `if (argc!=1)`
- (b) `if (argc<=0 || argc>=2)`
- (c) `if (argc!=2)`
- (d) `if (argv[]!=1)`
- (e) `if (argv[]!=2)`

5. (2 pts) What should be put in place of ^^^^?

- (a) `char argv[]`
- (b) `char *argv[]`
- (c) `char *argv`
- (d) `argv[0]`
- (e) `argv[1]`

6. (2 pts) Which of the following could be the first line of the main function in this program?

- (a) `int main(void)`
- (b) `int main(int argc, char argv[])`
- (c) `int main(int argv, char argc[])`
- (d) `int main(int argc, char *argv[])`
- (e) `int main(int argv, char *argc[])`

7. (2 pts) Now assume that the correct number of command line arguments was passed. Suppose that you have a variable `celsiusTemp` that will store the temperature passed in on the command line.

If `celsiusTemp` is declared of type `int`, which of the following correctly initializes this value from the command line argument?

- (a) `int celsiusTemp;`
- (b) `double celsiusTemp;`
- (c) `celsiusTemp = argv[1];`
- (d) `celsiusTemp = atoi(argv[1]);`
- (e) `celsiusTemp = atof(argv[1]);`

Number Conversions

8. (2 pts) Convert 91 from decimal to binary
- (a) 1001 0001
 - (b) 0101 1011
 - (c) 0101 1000
 - (d) 0001 1111
 - (e) none of the above
9. (2 pts) Convert FC18 from hexadecimal to binary
- (a) 1111 1100 0001 1000
 - (b) 1100 1111 1000 0001
 - (c) 1110 1100 0001 1000
 - (d) 1111 1100 1000 0001
 - (e) none of the above
10. (2 pts) Convert 0101 0111 from binary to decimal
- (a) 57
 - (b) 67
 - (c) 77
 - (d) 87
 - (e) none of the above

C Language Topics

11. (2 pts) Which of the following is a legal function prototype?

- (a) `int func(a);`
- (b) `int func(int [a]);`
- (c) `int func(int a[]);`
- (d) `void func(int nums[], size);`
- (e) none of these

12. (2 pts) Which of the following could be a legal function call?

- (a) `func(a);`
- (b) `int func(b[5]);`
- (c) `int func(int a[]);`
- (d) `void func(int nums[], size);`
- (e) none of these

13. (2 pts) In which of the following is x a *formal parameter*?

- (a) `printIt(x);`
- (b) `a = (int) x % b;`
- (c) `int daFunc(int x);`
- (d) `x = daOtherFunc(y);`
- (e) none of these

14. (2 pts) In which of the following is x an *actual parameter*?

- (a) `printIt(x);`
- (b) `a = (int) x % b;`
- (c) `int daFunc(int x);`
- (d) `x = daOtherFunc(y);`
- (e) none of these

File input and output

Suppose you have an input file called `data.dat`. Each line in is supposed to contain an integer value. Your task is to read in all of these lines of input, and find the sum and write that value out to a file called `report.txt`.

15. (2 pts) Read this question carefully: Which of the following *declares a variable* that could be used to access data currently stored in `data.dat`?
 - (a) `FILE *input;`
 - (b) `FILE *data.dat;`
 - (c) `input = fopen("data.dat", "r");`
 - (d) `input = fopen("data.dat", "w");`
 - (e) none of the above

16. (2 pts) Which of the following opens the file `data.dat` so that the program can do operations such as `fscanf` and `fgets`?
 - (a) `FILE *input;`
 - (b) `FILE *data.dat;`
 - (c) `input = fopen("data.dat", "r");`
 - (d) `input = fopen("data.dat", "w");`
 - (e) none of the above

17. (2 pts) Which of the following evaluates to true when there was a problem opening the file (e.g. `data.dat` doesn't exist).
 - (a) `if (input!=NULL)`
 - (b) `if (input==NULL)`
 - (c) `if (input==EOF)`
 - (d) `if (input!=EOF)`
 - (e) none of the above

18. (2 pts) Which of the following opens the file `report.txt` so that the program can do `fprintf` operations?

- (a) `FILE *outfile;`
- (b) `FILE *report.txt;`
- (c) `outfile = fopen("report.txt", "r");`
- (d) `outfile = fopen("report.txt", "w");`
- (e) none of the above

19. (2 pts) Suppose a variable has been declared as:

```
int x;
```

Which of the following reads an integer from the file into this variable?

- (a) `scanf("%d", x);`
- (b) `scanf("%d", &x);`
- (c) `fscanf(input, "%d", x);`
- (d) `fgets(x, 40, stdin);`
- (e) none of the above

What does it print?

Consider the following nested loop:

```
/* Gytha Ogg, Sample Program for CISC105 */  
  
#include <stdio.h>  
  
int main(void)  
{  
    int i, j = 2;  
    for (i = 0; i < 3; i++){  
        for (j = 0; j < 3; j++){  
            if (j < i)  
                printf("_");  
            else  
                printf("*");  
        }  
        printf("\n");  
    }  
    return 0;  
}
```

20. (5 pts) What does it print?

- (a) ___ (b) *** (c) *** (d) *___ (e) none of these
___* **_ _** **_
_** *___ ___* ***

Unix Commands

Consider the following script of a session on `strauss`. Note that three of the Unix commands have been replaced by the words in italics: such as *1st-missing-cmd*, *2nd-missing-cmd*, and *3rd-missing-cmd*. The three multiple choice questions that follow ask you what should be filled into each of those blanks.

```
> cd
> cd test
> pwd
/home/usra/d9/55560/test
> ls
lab01.c lab02.c
> 1st-missing-cmd
> ls
lab01.c lab02.c lab03.c
> 2nd-missing-cmd
> ls
lab02.c lab03.c tryit.c
> 3rd-missing-cmd
rm: remove lab03.c (yes/no)? y
> ls
lab02.c tryit.c
>
```

21. (3 pts) *1st-missing-cmd* should be replaced with:

- (a) `rm lab02.c lab03.c`
- (b) `dl lab02.c lab03.c`
- (c) `cp lab02.c lab03.c`
- (d) `mv lab02.c lab03.c`
- (e) `rn lab02.c lab03.c`

22. (3 pts) *2nd-missing-cmd* should be replaced with:

- (a) `rm lab01.c tryit.c`
- (b) `dl lab01.c tryit.c`
- (c) `cp lab01.c tryit.c`
- (d) `mv lab01.c tryit.c`
- (e) `rn lab01.c tryit.c`

23. (3 pts) *3rd-missing-cmd* should be replaced with:

- (a) `rm lab03.c`
- (b) `cd lab03.c`
- (c) `cp lab03.c`
- (d) `mv lab03.c`
- (e) `dl lab03.c`

What is the output?

On the following page is a sample program. Below is a script of the output from this program, with the numbers that it prints replaced by blanks.

First, trace through the program to determine what goes in each of these blanks, then select the correct answers from the choices given below.

Don't panic: Each of the sections of code can be done independently of the other sections, so if you are stuck on one, don't stop; keep working.

```
> ./q10
y=____
v=____ z=____
t=____
e=____ f=____
>
```

24. (3 pts) y=____
(a) y=-1 (b) y=-9 (c) y=9 (d) y=-8 (e) error
25. (3 pts) v=____ z=____
(a) v=5.1 z=5.1 (b) v=5.2 z=5.2 (c) v=5.2 z=5.1 (d) v=5.1 z=5.2 (e) error
26. (3 pts) t=____
(a) t=3 (b) t=5 (c) t=8 (d) t=15 (e) error
27. (3 pts) e=____ f=____
(a) e=5 f=4 (b) e=4 f=5 (c) e=5 f=5 (d) e=4 f=4 (e) error

```

/* q10.c Agnes Nitt for CISC105 final exam */
#include <stdio.h>

void trouble(int *r);
double fire(double f);
int burn(int b, int *n);
void cauldron(int *c, int *d);

int main(void)
{
    int e=2, f=3, t = 4, y = 9;
    double v=5.1, z = 10.2;

    trouble(&y);
    printf("y=%d\n", y);

    z = fire(v);
    printf("v=%4.1lf z=%4.1lf\n",v,z);

    t = 3;
    burn(5, &t);
    printf("t=%d\n",t);

    e=4;
    f=5;
    cauldron(&e,&f);
    printf("e=%d f=%d\n", e, f);
    return 0;
}

void trouble(int *r)
{
    (*r) = -1;
    return;
}

void cauldron(int *c, int *d)
{
    int temp = (*c);
    (*c) = (*d);
    (*d) = temp;
}

int burn(int b, int *n)
{
    (*n) = b;
}

double fire(double f)
{
    return (f + 0.1);
}

```

Programming

28. (20 pts) The main program listed below demonstrates a function that can put an array of temperature values in order from hottest to coldest. Sample output is given, along with a complete main program, and three function definitions. The last two function definitions on the following page are incomplete. Finish them, using the hints provided. Be extra careful with the order of parameters.

Sample output:

```
> ./a.out
Temps: 45.0 32.0 54.0 8.5 19.1
Hottest to Coldest: 54.0 45.0 32.0 19.1 8.5
>
```

Code:

```
/* Esmerelda Weatherwax, CISC105, Fall 2004 TA:Jason Ogg */
/* Demonstrate call to function that puts temperatures in order */
/* from hottest to coldest */

#include <stdio.h>

void hottestToColdest(int howMany, double temps[]);
void putColdestAtEnd(double tempArray[], int numTemps);
double outputArrayOnOneLine(int count, double array[]);

int main(void)
{
    double tempReadings[5]={45.0, 32.0, 54.0, 8.5, 19.1};

    /* output the temperatures before the function call */

    printf("Temps: ");
    outputArrayOnOneLine(5, tempReadings);
    printf("\n");

    /* put temperatures in order */

    hottestToColdest(5, tempReadings);

    /* output the temperatures after the function call */

    printf("Hottest to Coldest: ");
    outputArrayOnOneLine(5, tempReadings);
    printf("\n");

    return 0;
}
```

```

double outputArrayOnOneLine(int count, double array[])
{
    int i;
    for (i=0; i<count; i++)
        printf("%5.1lf ",array[i]);
}

void hottestToColdest(int howMany, double temps[])
{
    /* Hint: use a for loop, and a call to putColdestAtEnd.

    For you to decide: does the call go inside or outside
    the for loop? */

}

void putColdestAtEnd(double tempArray[], int numTemps)
{
    /* Hint: use a loop to find the index of coldest temperature,
    then swap that element with the last one in the array */

    int j;
    double temp;
    int indexOfColdest = 0; /* index of coldest element */

}

```

Debugging

To convert from fahrenheit to celsius, you subtract 32, then multiply by $\frac{5}{9}$. The code at the bottom of this page is “supposed” to convert from fahrenheit to celsius. Instead, it always prints

```
temp in celsius = 0.000000
```

as the result.

29. (3 pts) The program can be fixed by changing one line of code. Find this one line of code, cross it out, and write in the correct line of code.
30. (2 pts) Briefly explain why the original version of the program always prints 0.000000 as the answer.

(A one or two sentence explanation is sufficient; don't write a book!)

```
/* convert from fahrenheit to celsius */

#include <stdio.h>

int main (void)
{
    /* declare variables*/
    float faren, celsius;

    /* prompt user */
    printf("Enter a temperature in Farenheit: ");
    scanf("%f",&faren);

    /* calculate result and print */
    celsius = (faren - 32) * (5/9);
    printf("temp in celsius = %f \n",celsius);

    return 0;
}
```

Scoping Errors

31. (8 pts) Some lines in this program contain a particular kind of syntax error, specifically, a reference to a variable that is not "in scope". The error message from the compiler would say that the variable is "undeclared" or "undefined".

This is the only kind of syntax error in the following program.

You be the compiler: find each such line, and circle it.

You gain points for each line with an error that you find and circle.

You lose points each time you circle a line with no error.

There is at least one such error in this program, and there are no more than four.

```
/* q8.c  CISC105 Final Exam */
#include <stdio.h>

void doDatThing(int a, int b);
int doDaOtherThing(int j, int k);

void doDatThing(int a, int b)
{
    printf("a=%d b=%d\n", a, b);
    printf("a+b=%d\n", a+b);
}

int main(void)
{
    int x, y;
    int a;

    a = 8;
    x = 3;
    y = 2;
    doDatThing(x, y);

    printf("x=%d\n", x);
    printf("y=%d\n", y);
    printf("a=%d\n", a);
    printf("b=%d\n", b);
}

int doDaOtherThing(int j, int k)
{
    printf("j=%d\n", j);
    k = k+a;
    printf("k=%d\n", k);
}
```

End of Exam. Total Points: 100