CISC 105 sections @@@@@ Final Exam 12/15/04

Name_____

Please circle your section number:

010	014	018	022
011	015	019	023
012	016	020	024
013	017	021	025

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 @@@% multiple choice, and @@@% programming and short answer.

  The programming/short answer questions start with number **??**.
- Do problems you are confident about first. If you finish the problems you know, write what you do know about other problems to gain partial credit; but erroneous information may detract from that credit or irritate the grader, so don't make stuff up.
- Read *all* the directions *carefully* on each problem.
- Often writing a fast, rough version of a program in English or pseudocode will make your C coding faster and more accurate. It also enables me to give partial credit in some circumstances.
- You may assume that input will not produce errors for the procedures described, unless the question says otherwise.
- Do not do unnecessary testing. For example, testing for both `x < 0` and `x >= 0` instead of using one test and then `else` would be considered unnecessary testing.
- Good luck.

Multiple Choice: Mark answers on Scantron in #2 pencil

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 three nonsense symbols: `@@@@@`, `###` and `^^^^^`. (The questions which follow ask you to change these nonsense symbols to correct C code.)

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

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

1. (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)`

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

   (a) `%d` (b) `%c` (c) `%s` (d) `%f` (e) `%lf`

3. (2 pts) What should be put in place of ^^^^^?

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

4. (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[])`

5. (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]);`

What is the output?

The questions on the following page refer to this program:

```
#include <stdio.h>
/** Merv Mouthwash  CISC 105 **/

int changeOne(int k);
int changeTwo(int height, int weight);
int changeThree(int n);

int k = 3;
int g = 3;

int main(int x){

  int k = 2;
  int i = 1;
  printf("%d\n", changeOne(i));        /*AAAA*/
  printf("%d\n", changeTwo(i, g));     /*BBBB*/
  printf("%d\n", changeThree(g++));    /*CCCC*/
  printf("%d\n", changeOne(i));        /*DDDD*/
  return 0;
}

int changeOne(int k){
    return k + g;
}

int changeTwo(int j, int c){
    c++;
    return j + c;
}

int changeThree(int i){
  int c = 3;
  i += c;
  return i;
}
```

(These questions refer to code on the previous page)

6. (2 pts) The statement marked AAAA prints:

   (a) error (b) 1 (c) 2 (d) 3 (e) 4

7. (2 pts) The statement marked BBBB prints:

   (a) error (b) 3 (c) 4 (d) 5 (e) 6

8. (2 pts) The statement marked CCCC prints:

   (a) error (b) 3 (c) 4 (d) 5 (e) 6

9. (2 pts) The statement marked DDDD prints:

   (a) error (b) 3 (c) 4 (d) 5 (e) 6

Number Conversions

10. (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

11. (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

C Language Topics

12. (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

13. (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

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`.

14. (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

15. (2 pts) Read this question carefully: Which of the following *declares a variable* that could be used to place output from the program into `report.txt`?

    (a) `FILE *outfile;`
    (b) `FILE *report.txt;`
    (c) `outfile = fopen("report.txt","r");`
    (d) `outfile = fopen("report.txt","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. (4 pts) What does it print?

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

Strings

Answer the following questions based on a two-dimensional array of characters called `words` as follows:

char words[20][20] = ”I”, ”like”, ”the”, ”C”, ”language”;

21. (2 pts) Which of the following correctly prints the word “the”?

    (a) `printf('%c', words[2]);`
    (b) `printf('%s', words[3]);`
    (c) `printf('%s', words[2]);`
    (d) `puts(word, 10, stdout);`
    (e) none of these

22. (2 pts) Which of the following statements changes the letter “n” in “language”?

    (a) `char words[5][2] = 'A';`
    (b) `strcpy(words[5][2], "A");`
    (c) `strcpy(words[4][2], "A");`
    (d) `words[4][2] = 'A';`
    (e) none of these

23. (2 pts) Which of the following changes the word “like” to “love”?

    (a) `strcpy("like", "love");`
    (b) `strcmp(words[2][0], "love");`
    (c) `strcmp(words[2], "love");`
    (d) `strcpy(words[2][0], "love");`
    (e) none of these

---

24. (3 pts) Suppose you are to write a function like `strcmp`. Which of the following might be used as a prototype for that function?

    (a) `int myStrCmp(char a[], char b[]);`
    (b) `void myStrcmp(string a, string b);`
    (c) `int myStrcmp(string a, string b);`
    (d) `void myStrcmp(char a[], const char *b);`
    (e) none of the above

Unix Commands

Consider the following script of a session on strauss. Note that three of the Unix commands have been replaced by nonsense characters such as **@@@@@**, **#####**, and **%%%%%**. 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
> @@@@@
> ls
lab01.c  lab02.c  lab03.c
> #####
> ls
lab02.c  lab03.c  tryit.c
> %%%%%
rm: remove lab03.c (yes/no)? y
> ls
lab02.c  tryit.c
>
```

25. (3 pts) **@@@@@** 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`

26. (3 pts) ##### 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`

27. (3 pts) **%%%%%** should be replaced with:

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

Numeric Questions

Listed below are a series of code fragments. Assume that each of these appears in a complete C program, and that all necessary libraries have been included.

Your task: if the code fragment has no errors, select the answer that corresponds to the value of `x`. If the code fragment won't compile or will produce a run-time error, choose the answer "error". Each of the fragments is unrelated to the others; that is, "start from scratch with each question".

28. (2 pts)

```
float x,y;
y = 7;
x = floor(y/2);
```

(a) 3.0 (b) 3.5 (c) 4.0 (d) 7.0 (e) error

29. (2 pts)

```
int x,y;
y = 8/3;
x = ceil(y);
```

(a) 3 (b) 0 (c) 1 (d) 2 (e) error

30. (2 pts)

```
float x = 9 % 6;
```

(a) 3.0 (b) 54.0 (c) 1.0 (d) 2.0 (e) error

31. (2 pts)

```
int x, y=0;
for(x=0; x<100; x++)
    y++;
```

(a) 99 (b) 100 (c) 101 (d) 200 (e) error

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=___
>
```

32. (2 pts) `y=___`

    (a) y=-1 (b) y=-9 (c) y=9 (d) y=-8 (e) error

33. (2 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

34. (2 pts) `t=___`

    (a) t=3 (b) t=5 (c) t=8 (d) t=15 (e) error

35. (2 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

```c
/* 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);
}

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

36. (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 farenheit to celsius, you subtract 32, then multiply by $\frac{5}{9}$. The following code is "supposed" to convert from farenheit to celsius. Instead, it always prints

```
temp in celsius = 0.000000
```

as the result.

37. (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.

38. (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 farenheit 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);

}
```

Scoping Errors

39. (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: 109