CISC 105 sections 010-021 Final Exam Fall 2005

Name_____ Section_____ TA_____

General Instructions

- DO NOT WRITE YOUR NAME ON ANY PAGE EXCEPT THIS ONE!
- Turn off any noise making device, especially **CELL PHONES**. You may lose up to one letter grade if your device disturbs the peace of the exam.
- You have 50 minutes. **Pace yourself,** and pay attention to the point values.
- The exam has 25 multiple choice questions for 51% , and 49% programming.
- 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 questions say 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.
- You know a *lot*. Relax and focus on the problems at hand.

1. (2 pts) Which of the following would be a correct way to begin a main() that accepts command line arguments?

   (a) `int main(int argc, char *argv){`

   (b) `int main(int argc, char argv[]){`

   (c) `int main(int argc, char *pointers[]){`

   (d) `int main(int num, char *data()){`

   (e) none of the above

2. (2 pts) Which of the following function prototypes is the *best* choice for a function that prints a ”struct truck”?

   (a) void printTruck(struct truck data);

   (b) int printTruck(struct truck data[]);

   (c) void printTruck(struct truck *data);

   (d) int printTruck(struct truck *data[]);

   (e) none of the above

3. (3 pts) When structs are passed as parameters, they behave as

   (a) pass by name

   (b) pass by assignment

   (c) pass by value

   (d) pass by reference

   (e) none of the above

4. (3 pts) How many numbers could binary search look at to find one member of an array of 1,000,000 sorted numbers?

   (a) approximately 10

   (b) approximately 20

   (c) approximately 30

   (d) approximately 1,000,000

   (e) none of the above

5. (3 pts) How many numbers could linear search look at to find one member of a list of 1,000,000 numbers?

   (a) approximately 20

   (b) approximately 50

   (c) approximately 1,000

   (d) approximately 500,000

   (e) none of the above

6. (3 pts) What is wrong with this code fragment?

```
1)   int main(){
2)       char * w;
3)       strcpy(w, "spam");
```

   (a) w needs to be dereferenced in line 3

   (b) w is the wrong type for strcpy

   (c) w does not point to allocated memory

   (d) line 2 should have asterisk removed

   (e) none of the above

7. (3 pts) Which of the following is true?

   (a) a function's parameters are placed on the heap

   (b) malloc returns a char pointer

   (c) a function's local variables are placed on the stack

   (d) structs cannot be placed on the heap

   (e) none of the above

Pointers

name	address
x	fb0
y	fb4

```
int x = 4;
int y = 3;
int *p1, *p2;
```

8. (3 pts) To make `p1` point to `x`, use:

   (a) `*p1 = &x;`
   (b) `*p1 = *x;`
   (c) `p1 = *x;`
   (d) `p1 = {fb0};`
   (e) none of the above

9. (3 pts) Assume p1 points to `x`. To put 17 in `x`, use:

   (a) `*p1 += 17;`
   (b) `*x = 17;`
   (c) `*p1 = 17;`
   (d) `&p1 = 17;`
   (e) error

10. (3 pts) Assume p1 points to `x`. To make p2 point to `x`, use:

    (a) `*p2 = *p1;`
    (b) `p2 = *x;`
    (c) `p2 = &p1;`
    (d) `p2 = &x;`
    (e) error

11. (3 pts) Assume p1 points to `x`. To print the address of `x`, use:

(a) `printf("%p", p1);`
(b) `printf("%p", *x);`
(c) `printf("%d", &x);`
(d) `printf("%d", p1);`
(e) none of the above

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".

12. (2 pts)

```
x = 5 % 9;
```

(a) `-4` (b) `4` (c) `5` (d) `error` (e) none of the above

13. (2 pts)

```
x = 21 % 3;
```

(a) `0` (b) `3` (c) `7` (d) `error` (e) none of the above

14. (2 pts)

```
x = 3 > 3;
```

(a) `1` (b) `0` (c) `3` (d) `seg fault` (e) other error

15. (2 pts)

```
x = 18;
x /= 2;
```

(a) `16` (b) `8` (c) `9` (d) `error` (e) none of the above

16. (2 pts)

```
y = 7.2;
x = y * 2;
```

(a) `15` (b) `14.4` (c) `7` (d) `error` (e) none of the above

17. (2 pts)

```
double z = 15;
x = z / 10.0;
```

(a) `1.5` (b) `1` (c) `0.75` (d) `error` (e) none of the above

18. (3 pts) Which of the following expressions evaluates to the same answer as `!(x && y)` for all values of integers x and y?

(a) `!(x !|| y)`

(b) `(x !&& y)`

(c) `!(x || !y)`

(d) `(!x || !y)`

(e) the expression is unique

19. (3 pts) What is the output?

```
#include <stdio.h>
int main(void)
{
  int a[4] = {31, 4, 51, 11};
  printf("%d",a[4]);
  return 0;
}
```

(a) `0`

(b) `11`

(c) `4`

(d) `51`

(e) error

20. (3 pts) What is the output?

```
#include <stdio.h>
int main(void)
{
  int a[3] = {1,2};
  printf("%d",a[2]);
  return 0;
}
```

(a) `0`

(b) `2`

(c) `3`

(d) `NULL`

(e) error

21. (3 pts) What is the last index of the array `s` that is initialized after the declaration shown?

```
char s[10] = "espresso";
```

(a) `7`

(b) `8`

(c) `9`

(d) `10`

(e) `11`

22. (3 pts) Which expression could be correctly used to change the contents of c?

```
int main(){
    char c[4];
    ...
```

(a) `c = "cat";`

(b) `c = 'c';`

(c) `strcpy(c, "s");`

(d) `c = strcpy(c, "cats");`

(e) none of the above

23. (10 pts) Fill in the blanks. Be careful that your parameter names match the names used in the body of the function definition.

```
/*
 * Recursive function for searching an integer array sorted
 * small (left) to large (right).
 * Returns -1 if key is not found.
 */
int binarySearch(_____________________________________________
{

    int mid = _____________________________________________


    if (_____________________________________) return -1;

    if (data[mid] < key)

        return ____________________________________________

    else if  (data[mid] > key)

        return ____________________________________________
    else return mid;
}
```

24. (10 pts) Fill in the blanks. Be sure to look at the local variables that you need to work with, and remember to initialize them all. The problem focuses on determining exactly how much memory you need, then allocating that amount only. Keep in mind what you need to allocate space on the heap, and what you get back when you do it. Draw yourself some examples!

```
#include <stdio.h>

/* Program to demonstrate dynamic allocation */

char* makeWordSpace();

int main(){

    char *word = makeWordSpace();
    printf("Enter your word: ");
    /* put user input into new space */

    2 pts_____

    /* display word from new space */

    1 pts_____
}

/*
 * Allocates exactly the amount of memory that the user will
 * need. Prompts user for length of string to be entered.
 */
char* makeWordSpace(){

    int size;
    printf("How long is your word?\n");

    2 pts_____


    return  5 pts_____
}
```

25. (12 pts) Fill in the blanks. Read the program carefully and be sure to use the declared variables correctly. Be sure that all variables are initialized before you try to use their values.

```
#include <stdio.h>

/*  Program to keep track of the largest positive number
 *  entered by a user until user enters a negative number.
 *  Displays the result.
 */

int main(){

    double max;
    double data;

    printf("Enter a positive number, or a negative one to quit: ");
    scanf("%lf", &data);

    _____________________________________________


    while (_____________________________________________
        printf("Enter a positive number, or a negative one to quit: ");
        scanf("%lf", &data);

        if (_____________________________________


            _____________________________________
    }

    if (_____________________________________
        printf("No data was entered.\n");
    else
        _____________________________________

    return 0;
}
```

Structs

Use this definition for the following questions to write **ONE LINE** code answers for the following.

```
struct student{
    char name[30];
    int id;
};
```

26. (2 pts) Create a data structure to hold one student, and initialize it to id number 7 and a name of Martin.

27. (2 pts) Make a second student that holds the same values as the first.

28. (2 pts) Declare an array to hold data for 13,000 Udel undergrads.

29. (2 pts) Give the first student in the array the id number 26.

30. (2 pts) Give the first student in the array the name Puffin.