Name _____ Login name _____

Section: _____ TA _____

General Instructions

- DO **NOT** PUT YOUR **SSN** ON ANYTHING!
- 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 120 minutes for a 70 minute exam. Relax, **pace yourself,** and pay attention to the point values.
- The exam is 58 pts multiple choice, and 72 pts programming and short answer.
- Do not add features that are not required by the problem. For example, if the instructions don't say anything about user input, then your program should not take user input. If you aren't sure, ask.
- 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.
- Have fun!

Error definition

Errors do not always create problems in output, but should be considered errors anyway, as we do in class. This means errors include, but are not limited to:

- incorrect format specifiers
- incorrect number of parameters
- reading/writing invalid memory
- reading uninitialized ("garbage") memory
- other compilation errors
- other runtime errors

Errors do not include type coercion in arithmetic (for example, assigning a double to an int) unless that impairs the correctness of a program.

1. (2 pts) What is the output?

```
#include <stdio.h>
int main(void)
{
  int a[5] = {3, 4, 5, 6};
  printf("%d",a[4]);
  return 1;
}
```

(a) 0 (b) 1 (c) 6 (d) none of these (e) error

2. (2 pts) What is the output? (62 S06)

```
#include <stdio.h>
int main(void)
{
  int a[5] = {5, 6, 7, 8, 9};
  printf("%d",a[5]);
  return 0;
}
```

(a) 0 (b) 9 (c) 10 (d) none of these (e) error

3. (2 pts) What is the output? (75 S06)

```
#include <stdio.h>
int main(void)
{
  int a[6] = {1};
  printf("%d",a[3]);
  return 0;
}
```

(a) 1 (b) 6 (c) 3 (d) none of these (e) error

4. (2 pts) What is the output? (97 S06)

```
#include <stdio.h>
int main(void)
{
  int a[6] = {13, 15, 17, 19, 20, 21};
  printf("%d",a[3]);
  return 0;
}
```

(a) 6 (b) 20 (c) 17 (d) 19 (e) error

For the next five questions, if the code completes without error, select the answer that shows what is printed. If the code has an error (see the front page for definition of an error) select error. If no other answer is correct, select none of the above. Assume the variables will have the following addresses:

name	address
x	FB00
y	FB04
p	FB08

5. (2 pts)

```
int x = 2;
int *p = 5;
printf("%d\n", x);
```

(a) 2

(b) 5

(c) 0

(d) error

(e) none of the above

6. (2 pts)

```
int x = 5;
int *p = &x;
printf("%d\n", *p);
```

(a) x

(b) 5

(c) 0xFB00

(d) error

(e) none of the above

7. (2 pts)

```
int x = 5;
int *p = &x;
int y = *p;
printf("%d\n", y);
```

(a) x

(b) 5

(c) 0xFB00

(d) error

(e) none of the above

8. (2 pts)

```
int x = 2;
int n;
n = &x;
*n = 5;
printf("%d\n", x);
```

(a) 2
(b) 5
(c) 0
(d) error
(e) none of the above

9. (2 pts)

```
int x = 5;
int *p = &x;
*p = 2 * *p;
printf("%d\n", x);
```

(a) 2
(b) 5
(c) 10
(d) error
(e) none of the above

For the next **four** questions, assume x and y are declared as integers. Choose the letter that shows the value that would be in the variable x. Choose "error" if you think a fragment will not compile and run. The fragments are unrelated (each question is separate from the others).

10. (2 pts)

```
x = 0 || -1;
```

(a) `0` (b) `-1` (c) `true` (d) `error` (e) none of the above

11. (2 pts)

```
x = 4 % 5;
```

(a) `1` (b) `0` (c) `0.8` (d) `4` (e) none of the above

12. (2 pts)

```
x = 17 % 3;
```

(a) `1` (b) `2` (c) `3` (d) `error` (e) none of the above

13. (2 pts)

```
y = 5 == 6;
x = -7 && !y;
```

(a) `0` (b) `1` (c) `-7` (d) `7` (e) error

14. (2 pts) What is the last index of `w` that is initialized after the declaration shown?

```
char w[9] = "capybara";
```

(a) 7
(b) 8
(c) 9
(d) none of the above
(e) error

15. (2 pts) What is the last index of `w` that is initialized after the declaration shown?

```
char w[7] = "big rat";
```

(a) 2
(b) 3
(c) 7
(d) 8
(e) error

16. (2 pts) What is the output?

```
#include <stdio.h>
int main(void)
{
  char a[40][10] = {"albatross","spam","nugent","that"};
  printf("%c",a[1][3]);
  return 0;
}
```

(a) g
(b) m
(c) e
(d) none of the above
(e) error

17. (2 pts) Which of the following is a correct way to change c *after* the code shown?

```
int main(){
    char c;
    ...
```

(a) `char c = "s";`
(b) `c = "s";`
(c) `c = 's';`
(d) `strcpy(c, "s");`
(e) none of the above

18. (2 pts) Which of the following assigns a random value to x?

    (a) `rand(x);` (b) `x=rand();` (c) `random(x);` (d) `srand(x)` (e) none of these

19. (2 pts) Assume x contains a random number. Which of the following puts x in the range 50-100, inclusive?

    (a) `x = x % (100 - 50 + 1) + 50;`
    (b) `x = 50 + x % 50;`
    (c) `x = x % 50 + 51;`
    (d) `srand(x, 51, 50);`
    (e) none of these

20. (2 pts) How many numbers will binary search examine to find one member of a sorted list of 100,000 numbers?

    (a) approximately 6
    (b) approximately 16
    (c) approximately 17
    (d) approximately 21
    (e) none of the above

21. (2 pts) What is wrong with this code fragment?

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

    (a) w is the wrong type for strcpy
    (b) w does not point to allocated memory
    (c) line 2 should have asterisk removed
    (d) w needs to be dereferenced in line 3
    (e) none of the above

22. (2 pts) Which of the following is true?

    (a) a function's parameters are placed on the heap
    (b) malloc returns a void pointer
    (c) structs cannot be placed on the heap
    (d) pointer variables are placed on the heap
    (e) none of the above

23. (2 pts) Selection sort works in a series of passes over an array. Choose the answer that shows how this array will appear after the first TWO passes of selection sort (as discussed in class). Sort larger elements to the right.

    original array: 3 2 7 1 9 42 6 20

    (a) 2 1 3 7 6 9 20 42
    (b) 2 3 1 7 9 6 42 20
    (c) 42 20 3 2 7 1 9 6
    (d) 3 2 7 1 9 6 20 42
    (e) none of the above

24. (2 pts) Selection sort works in a series of passes over an array. Choose the answer that shows how this array will appear after the first TWO passes of selection sort (as discussed in class). Sort larger elements to the right.

    original array: 3 2 7 1 9 2 6 20

    (a) 3 2 7 1 9 2 6 20
    (b) 3 2 7 1 6 9 20
    (c) 20 9 3 2 7 1 2 6
    (d) 2 1 3 2 6 7 9 20
    (e) none of the above

25. (2 pts) Structs and arrays can be passed as parameters. When they are passed to a function, which can be modified in the *calling* function?

    (a) arrays, but not structs
    (b) both structs and arrays
    (c) structs, but not arrays
    (d) neither can be modified
    (e) neither can be passed as parameters

26. (2 pts) When a struct is passed as a parameter to a function using a function call, C uses

    (a) pass by assignment
    (b) pass by reference
    (c) pass by name
    (d) pass by value
    (e) none of the above

    Consider the following declaration when answering the next three questions.

```
typedef struct {
    int legs;
    double weight;
    char name[100];
} Rat;
```

27. (2 pts) Assume two such data structures, **ben** and **willard**, have been declared, and **willard** has been initialized[1]. Which of the following will take the values from willard and put them into ben?

    (a) willard->ben;
    (b) strcpy(willard, ben);
    (c) strcpy(ben, willard);
    (d) it must be done with a while loop
    (e) none of the above

28. (2 pts) Which of the following functions could take one of the structs shown above as a parameter?

    (a) void print(Rat k);
    (b) void print(struct Rat k);
    (c) void print(Rat k[]);
    (d) Rat print(typedef Rat k);
    (e) none of the above

[1] Who will sing to Ben?

29. (2 pts) Assume you have correctly made a new struct of the type above, named `x`, so that x is a struct. Which of the following correctly gives `x` a new weight?

    (a) struct x.weight = 6;

    (b) struct x->weight = 6;

    (c) x->weight = 6;

    (d) error

    (e) none of the above

Short Answers

Consider the **partial** program below. Imagine that you are going to add some functions to it. Make up names as necessary, and use parameters from the existing code in your calls.

```
#include <stdio.h>

/* Terry Harvey CISC105-20 TA: Liric Waterchard*/

int main(){

    int someData[100] = {1,2,3,4,5,6,7};
    int one = 5;
    ...
```

30. (3 pts) Write a prototype for a function that can change any element in someData. Do not write the definition.

31. (3 pts) Show how you could call the function from number 30 in main().

32. (3 pts) Write prototype for a function that can change the value in the variable "one" declared in main.

33. (3 pts) Show how you could call the function from number 32 in main() so that it can alter the value in "one".

34. (3 pts) What is the maximum number of elements a binary search would have to examine before it could determine that the key was not present in an array of 800 elements?

35. (3 pts) Show the binary equivalent of the decimal number 254. You must show your work.

36. (3 pts) When we discussed the idea of a linked list of structs in class, we discussed one algorithm that we can't implement efficiently with a linked list. What was the algorithm?

37. (6 pts) Suppose we have the declarations

```
char data[20];
char * temp;
```

and then we put a string into data. Now show how to set temp to point to new space that you allocate, just big enough to hold whatever is in data.

38. (15 pts) Read this program and its comments carefully, then fill in the blanks neatly. Do not change what I have written if you want full credit. ONLY fill in the blanks. Use any function you need based on its prototype and description (and your knowledge of the sorting algorithm we wrote in class).

```
#include <stdio.h>
#define SIZE 6
/* Terry Harvey  CISC 105 Section: 00  TA: Sprenkle */

/* Code to demonstrate selection sort on the final. Using functions
   whose prototypes are provided*/

/* Performs selection sort recursively; calls other functions to do work. */
void selectionSort(int a[], int size);

/* Prints an integer array */
void printArray(int a[], int size);

/* Given an index, swaps the number at that index with the number at
   the rightmost end of the array. */
void putElementAtEnd(int allElements[], int size, int index);

/* Finds the largest element in an array */
int findLargestElement(int a[], int size);

/* Locates key in an ordered integer array */
int binarySearch(int data[], int key, int start, int end);

/* Calculates average of an integer array */
double average(int data[], int size);

int main(){

    int data[SIZE] = {3,42,2,7,1,9};
    printArray(data, SIZE);
    selectionSort(data, SIZE);
    printArray(a,SIZE);

    return 0;
}

/* Fill in the blanks. Think before you code! Do not use more than the
   lines provided to receive full credit. Do not write other
   functions.*/

void selectionSort(int a[], int size){
    if (size == 1) return;

    int maxIndex;

    /*5 pts*/_____

    /*5 pts*/_____

    /*5 pts*/_____

    return;
}
```

39. (15 pts) Read this program and its comments carefully, then fill in the function neatly. Do not change what I have written if you want full credit. ONLY fill in the function body.

```
#include <stdio.h>

/* Terry Harvey CISC105 Section 21 TA Aelan Jinblan-Waschrag*/

/* Sample program for final */

int countChar(char input, char data[]);

int main(){
    char word[20] = "aspaamallama";
    int count = countChar('a', word);

    printf("%d\n", count);

    return 0;
}

/*
 * This function counts the number of times the char parameter
 * occurs in the data string; returns the count.
 * FILL IN THIS FUNCTION. THIS ONE RIGHT HERE.
 */
int countChar( char input, char data[] ){
    int i;
    int count = 0;









}
```

40. (15 pts) Read this program and its comments carefully, then fill in the blanks neatly. Do not change what I have written if you want full credit. ONLY fill in the blanks.

```
#include <stdio.h>

/* Terry Harvey CISC105 Section TA*/

/* Final final program */

typedef struct {
    char name[30];
    int teeth;
} Pet;

int main(){

    Pet ted;

    Pet *tedPtr;

    Pet zoo[10];

    /* Give ted the name Jill and 7 teeth */

    _____________________________

    _____________________________


    /* Show how to get all of ted's data into the first element of zoo
       in one statement.*/

    _____________________________

    /* Make tedPtr point to ted */

    _____________________________


    /* Use tedPtr to change ted's teeth to 8 */

    _____________________________


    return 0;
}
```