**QUESTION 1**

1. In the code:

char \* cp;

cp has size (give the number of bytes or say **unknown**):



**5 points**

**QUESTION 2**

1. In the code:

struct point{

…..  // assume correct code here, but you do not know what it is

};

struct point temp1;

temp1 has size (give the number of bytes or say **unknown**):



**10 points**

**QUESTION 3**

1. In the code:

struct point{

…..  // assume correct code here, but you do not know what it is

};

struct point \* temp2;

temp2 has size (give the number of bytes or say **unknown**):



**10 points**

**QUESTION 4**

1. How many bytes are allocated by the malloc call highlighted in the code below?

struct point {   // line 1

    int x;       // line 2

    int y;       // line 3

    int z;       // line 4

};

void mem\_7(){            // line 5

    struct point \* p;    // line 6

    p = (struct point \*)**malloc( sizeof(struct point \*) )**;   // line 7

    p->x = 3;       // line 8

    p->y = 7;       // line 9

    p->z = 0;       // line 10

    free(p);        // line 11

}



**10 points**

**QUESTION 5**

1. Given the program:

void test1(int v){

    v = 20;

    printf("test 1: v = %d\n", v);

}

void test2(int \* v1){

    \*v1 = 10;

    printf("test 2: v1 = %d\n", \*v1);

}

void test3(int \* v1){

    v1 = (int\*)malloc(sizeof(int));

    \*v1 = 15;

    printf("test 3: v1 = %d\n", \*v1);

}

int main(int argc, char\*\* argv) {

    int v = 0;

    printf("main, 1, v = %d\n", v);

    test1(v);

    printf("main, 2, v = %d\n", v);

    int\* v1;

    v1 = (int\*)malloc(sizeof(int));

    \*v1 = 5;

    printf("--------------------\n");

    printf("main, 3, v1 = %d\n", \*v1);

    printf("main, 4, v1 = %d\n", v1);

    printf("--------------------\n");

    test2(v1);

    printf("main, 5: v1 = %d\n", \*v1);

    printf("--------------------\n");

    test3(v1);

    printf("main, 6: v1 = %d\n", \*v1);

    free(v1);

    return 0;

}

In the questions below are displayed some of the lines that the program will print. Fill in the blanks for each line with what the program prints on that line. (Some lines that the program prints are skipped. Do not worry about them.)

If it will print a memory address, answer with the word **memory**.

If it print a value that is unknown (e.g. from an uninitialized variable), answer with the word **unknown .**

You should assume that uninitialized variables get random values.

main, 2, v = 



**10 points**

**QUESTION 6**

1. main, 3, v1 =



**5 points**

**QUESTION 7**

1. main, 4, v1 =



**5 points**

**QUESTION 8**

1. main, 5: v1 =



**5 points**

**QUESTION 9**

1. main, 6: v1 =



**10 points**

**QUESTION 10**

1. int \* mem\_1\_in(int x){  // line 1

    x = x + 2;          // line 2

    return &x;          // line 3

}

void mem\_1(){           // line 4

    int x = 10;         // line 5

    int \* p = mem\_1\_in(x);        // line 6

    printf("\*p = %d\n\n", \*p);    // line 7

}

Where are function parameters saved in C? On the ***heap*** or the ***stack***? 

Look at line 3 from the code above. Is it *safe* to return a pointer to the parameter x, or is it an *error*? (For your answer say ***safe*** or ***error***) 

**10 points**

**QUESTION 11**

1. This and the following questions provide code and you have to answer whether or not there is a bug in that code.

If yes, give the number of the line where the bug is.

If there are bugs in more than one line or the code is wrong because of a combination of lines, list all these line numbers in increasing order, separated with a comma and with no extra spaces (e.g. 1,4,6,7).

If the code is wrong because there is a memory leak (allocated memory was not freed) answer with the word **leak** .

If the code is correct (no bug), answer with the word **none** .

Example of correct answer formats:

3

1,5,6,7

none

leak

Examples of incorrect answer formats:

5,6,7,1   (not in increasing order)

1 5 6 7   (separated by spaces, not by commas)

/\* Intended behavior for this function:

Allocate space on the heap, write 25 in that space, print and free up the space. Variable p is the pointer used to access that space (the pointer to that space).

\*/

int\* mem\_3(){         // line 1

    int \*p = 25;      // line 2

    printf("p = %d\n", \*p);    // line 3

    return p;          // line 4

}

Are there any errors in this function? Give the line(s) with bugs or **none** .



**5 points**

**QUESTION 12**

1. /\* The intended behavior for this function is to:

Allocate space on the heap, write 7 in that space, print and free up the space. Variable p is the pointer used to access that space (the pointer to that space).

\*/

void mem\_5(){   // line 1

    int \* p;    // line 2

    p = (int\*)malloc(sizeof(int));   // line 3

    p = 7;      // line 4

    free(p);    // line 5

}



**5 points**

**QUESTION 13**

1. /\* This function manipulates local variables.

Do not worry if what it does makes sense. (Do not worry about logical errors.)

***Look only for memory-related errors*** (e.g. invalid memory access, memory leak,...)

\*/

void mem\_4(){   // line 1

    double d;   // line 2

    double \*p;  // line 3

    p = &d;     // line 4

    \*p = 7;     // line 5

}

