IC221 System Programming
Spring 2014 NAME:__________________

HW4 COLLABORATOR(S):______________________

5/3/0 1. For the given code block below, **circle** all the variables that are allocated on the stack, and **box** all the variables that are allocated on the heap.

```
int fun(int a){
    int b = 10;
    int * c = (int *) malloc(sizeof(int));
}
```

7/5/3/ 2. Draw the stack diagram for the code sequence below at **POINT**.

```
int * decrement(int * p){
    *p -= 1;
    return p;
}

int main(int argc, char *argv[]){
    int * p, * q, a;
    a = 10;
    p = &a;
    q = decrment(p);
    /* POINT */
}
```

3.

What’s wrong with the following function?

7/5/3/0

```
int * fun(){
    int a;
    //...
    return &a;
}
```

6/4/2/0 4. When a function returns, why does the memory get deallocated automatically?

/25

NAME:__________________

7/5/3/0 5. Why does the stack and heap grow in alternate directions, stack down and the heap up?

6/5/3/0 6. Why is there a need to have both the stack and the heap?

7. For the code segment below, draw the **execution stack** at each push and pop starting with **main()**.

```
int add(int a, int b){
    return a+b;
}

int minusone(int a){
    return add(a,-1);
}

int times(int a, int b){
    return a*b;
}

int timestwo(int a){
    return times(a,2);
}

int main(int argc, char * argv[]){
    add(minusone(2), 3);
    return 0;
}
```

12/10/8/5/0

/25

NAME:__________________

8. For the following deceleration

```
int array[4];
int * p = array+2;
```

a. Draw the stack diagram: 8/5/3/0

5/3/0 b. What index of **array** is **p[1]**?

5/3/0 c. What index of **array** does ***(p-1)** dereference?

5/3/0 9. Write the **malloc()** command to allocate an array of 15 **doubles**.

```
double * darray =
```

7/5/3/0 10. What are the two problems of using **malloc()** to allocate arrays and how does **calloc()** address those problems?

/30

NAME:_________________

5/3/0 11. Write the type declaration for a variable **strings** that can store an array of C strings. Just type declaration, not allocation)

13. Below is a code segment with a double pointer allocation:

```
mytype_t ** mytypes = (mytype_t **)calloc(13, sizeof(mytype_t *))

mytypes[10] = (mytype_t *) calloc(1, sizeof(mytype_t));

//... Rest of program
//including other allocations to other indexes of mytpes
```

Assume that more allocations occurred in the rest of the program to any possible indexes of **mytypes**. Write the deallocation routine:

15/13/10/5/3/0

/20