IC221 System Programming
Spring 2015 NAME:_____
HW5

COLLABORATOR(S):_____

5/3/1/0 1. Identify the memory violation in the following program. **Circle it and describe it to the right.**

```
int * makearray(int size){
      int array[size];
      int j;
      for(j=0;j<size;j++){
            array[j] = j*2;

      return array;
}

int main(int argc, char * argv[]){
      int * a1 = makearray(10);
      int * a2 = makearray(10);
      int j, sum=0;

      for(j=0;j<10;j++){
            sum+=a1[j]+a2[j]
      }
      printf("sum: %d\n", sum);
}
```

5/3/1/0 2. For the above program, rewrite the makearray() function such that it does not have a memory violation

5/3/1/0 3. Explain how your corrected version of makearray() does not have the same memory violations.

5/3/1/0 4. When a function returns, why are the local stack variables de-allocated?

5/3/1/0 5. Why is there a need to have both a stack and a heap?

__/25

NAME: _____

5/3/1/0 6. Draw and label the program memory layout to the right. **Indicate which direction the stack and heap grows.**

5/3/1/0 7. What kind of memory typically exists between the stack and the heap?

10/8/4/2/0 8. For the code segment below draw the stack model, of pushes and pops of function frames, through the end of execution, i.e., main() being popped off the stack.

push main **|__main____|**

push minusone **| add |**
|__main____|

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


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

int sub(int a, int b){
      return a-b;
}

int main(){
      int i = times(add(1,2),5)
      sub(i,6);
}
```

__/20

NAME: _____

5/3/1/0 9. Using **malloc()** write the command to allocate an array of 16 long values:

```
long * larray = malloc (                                    );
```

10. Using **calloc()** write the command to allocate the same array of 16 long values:

```
long * larray = calloc(                                    );
```

5/3/1/0 11. What are the two differences between **malloc()** and **calloc()** with respect to array allocations?

15/13/10/5/0 12. Consider the following code sample for dynamically allocating an array of **mytype_t** structures. Fill in the function deallocate such that there are no memory leaks:

```
typdef struct{
      int * a;  //array of ints
      int size; //of this size
} mytype_t;

mytype_t * allocate(int n){
      int i;
      mytype_t * mytypes = calloc(n,sizeof(mytype_t*));
      for(i=0;i<n;i++){
            mytypes[i]->a = calloc(i+1,sizeof(int));
            mytypes[i]->size = i+1;
      }
      return mytpes;
}

void deallocate(mytype_t * mytypes, int n){

}
```

5/3/1/0 13. Explain your deallocate function above and why you **free()**'ed what you did

__/30

NAME: _____

5/3/1/0 14. Explain why this is a legal cast between pointer types:

```
int a = 10;
char * p = (char *) &a;
```

5/3/1/0 15. Continuing with the snippet code above, what does **p[2]** reference with respect to the integer **a**.

5/3/1/0 16. Consider the code snippet below that prints the bytes of the integer **a** in hexadecimal, what is the output?

```
unsigned int a = 0xcafebabe;
unsigned char * p = (char *) &a;
int i;

printf("0x");
for(i=0;i<4;i++){
     printf("%02x",p[i]);
}
printf("\n")
```

5/3/1/0 17. What is the difference between Big and Little Endian? **Use the above program output as part of your explanation.**

5/3/1/0 18. Which endian representation does most computers use? How can you tell from the sample program?

__/25