1. **Explain why an operating system can be viewed as a resource allocator. (10 points)**

A computer system has hardware and software that may be required to solve a problem, like CPU time, memory space, file-storage space, I/O devices, etc. The OS acts as a manager for these resources so it is viewed as a resource allocator.

1. **Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU’s execution load. (10 points)** 
   1. **a)  How does the CPU interface with the device to coordinate the transfer?**

The CPU can initiate a DMA operation by writing values into special registers that can be independently accessed by the device. The device initiates the corresponding operation once it receives a command from the CPU.

* 1. **b)  How does the CPU know when the memory operations are complete?**

Once the entire transfer is finished, the DMA controller interrupts the CPU.

* 1. **c)  The CPU is allowed to execute other programs while the DMA controller is transferring data. Does this process interfere with the execution of the user programs? If so, describe what forms of interference are caused.**

A problem would happen if both the CPU and the DMA controller want to access the memory at the same time. So, the CPU should be momentarily prevented from accessing main memory when the DMA controller seizes the memory bus. However, if the CPU is still allowed to access data in its primary and secondary caches, a coherency issue may be created if both the CPU and the DMA controller update the same memory locations.

1. **Describe the operating system's two modes of operation. (5 points)**

Kernel (0) or user (1) are the two modes of operation. When the computer system is executing on behalf of a user application, the system is in user mode. However, when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to fulfill the request.

1. Write a C program that counts the number of non white-space characters in an input text file. Program takes as command argument(s) the name of the input file (and the output file with option -f). Based on the option flags, it displays the output on the standard output, or writes the output to an output file. (25 points)

The command format is as follows:  
command -f inputfile outputfile

or,

command -s inputfile

-f indicates writing to an output file;  
-s indicates displaying the output on the screen.

Also notice the errors caused by incomplete arguments.

Be sure to document your program. Documentation should include: (1) The name of the author of the program (your name) and the date it was written, (2) a concise description of what the program does, (3) a clear description of its limitations if it does not meet all specifications, and (4) comments of critical code and data declarations. The same should also be done for each function written in the program. You’ll lose some points if the above documentation is missing.

/\*  Julio Quintero 11/9/2019 dd/mm/yyy

    This program is a simple code that will count the number of none white spaces in a text file. It does this by counting all the characters and substracting the white spaces.

    It is simple code, it has been tested on pdf, word, or other type of file that isn't a text file.

\*/

#include <stdio.h>

#include <stdlib.h>

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

    // Variables

FILE \*file;

*char* c,ch,total;

*int* char\_count = 0, count = 0;

// Check if file can be open

if ( (file = fopen(argv[1], "r")) == NULL) {

fprintf(stderr,"%s: cannot open %s \n", argv[0], argv[1]);

exit(1);

}

// Extract characters from file

    // and store in character c

    for (c = getc(file); c != EOF; c = getc(file))

        // Increment count for all characters

        char\_count++;

    //count white spaces

    while((ch = fgetc(file))!=EOF)

                  if(ch==' ')

                      count++;

// (character - white spaces) will give all none white characters

total = char\_count - count;

    // Print the count of characters

    printf("The file has %d characters\n ", total);

 // Close the file

    fclose(file);

    return 0;

}