AOS ASSIGNMENT 6

SOURCE CODE FILES LIST:

shprototypes.h	Urmil	Made changes for creating new command memmanage
shell.c	Riddhi	Made changes for creating new command memmanage
memmanage.c	Riddhi	The main file from which we examine how memory management works by creating variables and processes to check heap and stack storage as well as print the free memory information
printFreeMem.c	Urmil	Print a list of starting address of free memory locations and the block size available
own_create.c	Riddhi	Create the process by calling getownstk()
own_getmem.c	Urmil	Allocate space for the heap storage of variables in the free memory block
getownstk.c	Urmil	Allocate stack space for the process in the free memory block
proc_function.c	Urmil	Initialize a demo user defined function
proc_function.h	Riddhi	Define the header files which include attributes of process
own_kill.c	Riddhi	Checked for the used stack space before the process is killed and the memory is not freed.

EXERCISE 9.1

Write a function that walks the list of free memory blocks and prints a line with the address and length of each block.

DEMO OF FUNCTIONALITY:

memmanage.c is the main file through which we call the various memory management functions.
We call printFreeMem() to print the free memory space available, printing the starting location and the block size available.

EXERCISE 9.4

Replace the low-level memory management functions with a set of functions that allocate heap and stack memory permanently (i.e., without providing a mechanism to return storage to a free list). How do the sizes of the new allocation routines compare to the sizes of *getstk* and *getmem*?

COMPARISON OF SIZES

New(our implementation) allocation	Original allocation
getownstk: In this implementation we have implemented the first fit allocation for stack, that is, the first free block found while traversing the free memory list from the lowest memory location, will be allocated to the stack. We do not de-allocate space for stack storage. The disadvantage of this is that it can create holes or gaps in the memory when we explicitly free these allocated blocks.	getstk: In the standard XINU implementation, the stack follows the last fit policy, that is, the free memory space is traversed from the highest memory location and the first available free block from the end is allocated to the stack. getstk is used to store processes and their information.
own_getmem: Our implementation is similar to the original XINU implementation. We do not de-allocate space for heap storage	getmem: The memory for heap is allocated using getmem function to store variables. This memory is allocated using the first free block found from the lowest free memory blocks.

DEMO OF FUNCTIONALITY:

We create a variable which will allocate a space in heap memory thus changing the starting memory location since the heap allocation begins from the lowest memory location. Thus the starting location of free memory and the available block size changes.

EXERCISE 9.7

Many embedded systems go through a prototype stage, in which the system is built on a general platform, and a final stage, in which minimal hardware is designed for the system. In terms of memory management, one question concerns the size of the stack needed by each process. Modify the code to allow the system to measure the maximum stack space used by a process and report the maximum stack size when the process exits.

DEMO OF FUNCTIONALITY:

We create a process while will initialize a stack with the specified stack size. We then compute the total stack size allocated, the stack size used and the stack memory available after the process is killed manually.