Isabel Lally

Homework 2

1. Lists and Structures

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

struct procInfStruc{

  int processId;
  char *processContext[6];
  char processStatus;
  struct procInfStruc *prev;
  struct procInfStruc *next;

};
```

2. C Pointers and References

```
#include <stdio.h>
#include <memory.h>
#include <string.h>
#include <stdlib.h>

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

  // Checks for correct number of parameters
  if(argc != 2){
    printf("Requires 1 Parameter");
    exit(1);
  }

  int size = atoi(argv[1]);

  // Allocate block of memory the size of parameter.
  char *blockMem = malloc(size);

  memset(blockMem, 0, size);
  printf("The address of the memory block: %p\n", &blockMem);


  char *point = blockMem;

  for(int i = 0; i < size; i++){
    printf("Content %d: %x\n", (i+1), *point);
    point++;
  }

  // Free the memory
  free(blockMem);

}
```

2. Sample Output:

```
./mallocMem 4
The address of the memory block: 0x7ffe816a4e08
Content 1: 0
Content 2: 0
Content 3: 0
Content 4: 0
```

3. Process Transitions

Ready --> Suspended and Running

Ready may go to suspended if the scheduler decides that a different process had priority.

Ready will move to running if the process is executed.

Running --> Ready

Running to ready may happen if the process is waiting for the scheduler to decide what to do.

Suspended --> Ready

Suspended to ready will occur when there is room for the process to move into main memory.

4. Scheduling Policies

CFS

The algorithm for the Completely Fair Scheduler (CFS) uses time slices and is priority based. The time slices are fixed, so if the process is not completed in that time it is rescheduled. The higher priority processes are the ones that will take less time and thus end up at the front of the line. CFS uses scheduling domains where processes can be balanced between different processors and scheduling groups which groups processes together if they are similar.

BFS

The algorithm for the Brain Fucker Scheduler (BFS) uses priority classes, where processes running cannot be interrupted by lower priority processes. However, processes that end up in the ready state and have higher priority than the process currently running can interrupt that process.