/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*memoryManagement.c\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
\* **Programmer**:  Sarath Madala   
\*   
\* **Course**:  CSCI 4353.01   
\*   
\* **Date**:  February 05, 2018

\*   
\* **Assignment**:  Program : Memory Management  
\*   
\* **Environment**:  UNIX running on putty  
\*   
\* **Files Included**: a:\\program1.c,   a:\\testin.dat,   a:\\testout.dat   
\*   
\* **Purpose**:  Memory manager program which handles memory requests by determining starting \*address, final address, and the amount of internal fragmentation for the fixed sized partitions. \*The memory requests are made in first fit order   
\*   
\* **Input**:  5 integers are in c:\\path\testin.dat   
\*   
\* **Preconditions**:  Only integers between 0 to 9 are in the input file   
\*   
\* **Output**:  A sum is calculated and printed to a:\\testout.dat, or an error message is printed   
\*   
\* **Postconditions**:  
\*   
\* **Algorithm**:   
\*          get the number of partitions and size of partitions in struct  
\*          get memory requests from the user and store them in array  
\*          for(i=0; i<length(array); i++)  
\*           startingAdd = i\*size of partition;  
\*           finalAdd = (i\*size of partition)+array(i)-1;  
\*           totalFragmentation = totalFragmentation + (10 – array(i));  
\*           print startingAdd, finalAdd;  
\*          end loop  
\*           print totalFragmentation;  
\*            
\*            
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
#include <stdio.h>

#include <stdlib.h>

struct sizeandNumber{

int size;

int number;

}val;

void main(){

int i, n, startingAddress, finalAddress, totalFragmentation=0;

int \*request;

printf("\n Enter size of Partitions: ");

scanf("%d",&val.size);

printf("\n Enter number of Partitions: ");

scanf("%d",&val.number);

printf("\n Enter number of requests: ");

x: scanf("%d",&n);

if(n > val.number){

printf("\n Enter value less than %d", val.number);

goto x;

}

request = (int \*) malloc(n\*sizeof(int));

printf("\n Input the memory requests");

for(i=0;i<n;i++){

printf("\n Memory request %d = ", i+1);

scanf("%d", &request[i]);

}

for(i=0;i<n;i++){

startingAddress = i \* val.size;

finalAddress = (i\*val.size) + request[i]-1;

totalFragmentation += val.size-request[i];

printf("\n Request %d Starting Address = %d", i+1, startingAddress);

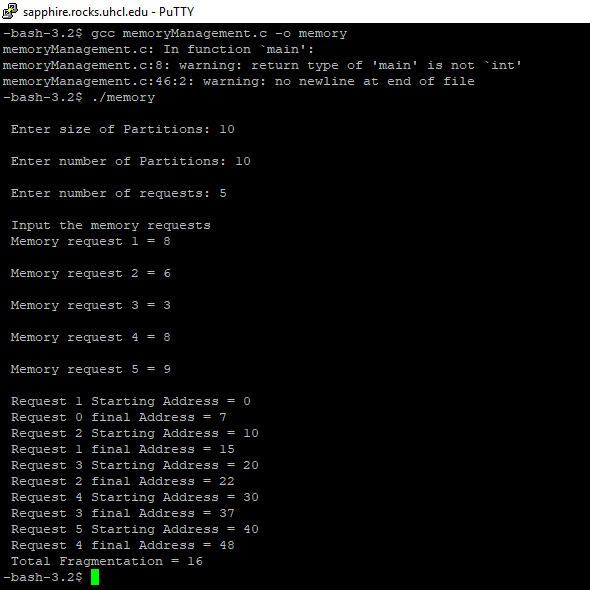
printf("\n Request %d final Address = %d", i, finalAddress);

}

printf("\n Total Fragmentation = %d \n", totalFragmentation);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Output:**

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/