231044

O.ESHWARAGE

SCS 1301

Exercise 6

2.

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

#define MEMOERY_SIZE 200
typedef struct BLock{
        int start;
        int size;
}BLock;

static unsigned char memory[MEMOERY_SIZE];
static BLock allocationTable[MEMOERY_SIZE];
static int allocationCount=0;

void* NewMalloc(int size);
void Newfree(void* ptr);
void DefragmentMemoryz();
void PrintMemory();

int main(){
        int *a=(int*)NewMalloc(sizeof(int));
        float *b=(float*)NewMalloc(sizeof(float)*3);
        char *c=(char*)NewMalloc(sizeof(char)*10);

        PrintMemory();

        Newfree(a);
        Newfree(b);

        DefragmentMemoryz();

        PrintMemory();

        return 0;
}

void* NewMalloc(int size){
```

```c
        if(allocationCount>=MEMOERY_SIZE){
                printf("Memory is full\n");
                return NULL;
        }

        int freeStart=-1;
        int freeSize=0;

        for(int i=0;i<MEMOERY_SIZE;i++){
                if(memory[i]==0){
                        if(freeStart==-1)freeStart=i;
                        freeSize++;
                        if(freeSize>=size){
                                for(int j=freeStart;j<freeStart+size;j++){
                                        memory[j]=1;
                                }
                                allocationTable[allocationCount++]=(BLock)(freeStart,size);
                                return &memory[freeStart];
                        }
                }else{
                        freeStart=-1;
                        freeSize=0;
                }
        }
        printf("Not enough contigous memory available\n");
        return NULL;
}

Void NewFree(void* ptr){
        unsigned char* p=(unsigned char*)ptr;
        int start=p-memory;

        for(int i=0;i<allocationCount;i++){
                if(allocationTable[i].start==start){
```

```
        for(int i=0;i<allocationCount;i++){
                if(allocationTable[i].start==start){
                        for(int j=start;j<start+allocationTable[i].size; j++){
                                memory[j]=0;
                        }
                        for(int j=i;j<allocationCount-1;j++){
                                allocationTable[j]=allocationTable[j+1];
                        }
                        allocationCount--;
                        return ;
                }
        }
        printf("Pointer not found in allocation table\n");
}

void DefragmentMemory(){
        int nextFree=0;
        for(int i=0;i<allocationCount;i++){
                BLock block=allocationTable[i];
                if(block.start !=nextFree){
                        memmove(memory+nextFree,memory+block.start,block.size);
                        allocationTable[i].start=nextFree;
                }
                nextFree+=block.size;
        }
}

void PrintMemory(){
        printf("Memory blocks:\n");
        for(int i=0;i<MEMOERY_SIZE;i++){
                printf("%d,memory[i]");
        }
        printf("\n");
}
```