ForNextDay(11)
Stephen Cole
3553803

Integer.h

```
#ifndef INTEGER_H
#define INTEGER_H

// a cover function for malloc()
// malloc and return memory for a int array of intsize
// return (int*)NULL on failure
int* mallocIntArray(int intsize);

// create a duplicate int array of iArr
// return it
// return (int*)NULL on failure
// should call mallocIntArray(), then copy
int* duplicateInt(int* iArr);

int** duplicateIntArrays(int** iArrs, int size1d, int size2d);

#endif
```

integerarray.c

```
#include"Integer.h"
#include<stdlib.h>
#include<stdio.h>

int* mallocIntArray(int intsize)
{
	return (int*) malloc(sizeof(int)*intsize);
}

int* duplicateInt(int* iArr)
{
	int* duplicate = mallocIntArray(sizeof(iArr));
	int iArrSize = sizeof(iArr)/2;
	int i;

	for(i=0; i<iArrSize; i++)
		duplicate[i] = iArr[i];

	return duplicate;
}
```

```
int** duplicateIntArrays(int** iArrs, int size1d, int size2d)
{
	int i;
	int *storage = (int*)malloc(size1d*size2d*sizeof(int));
	int **dup = (int**)malloc(size1d*sizeof(int*));

	for(i=0; i<size1d; i++)
		dup[i] = storage + 4*i;

	int j;
	for(i=0; i<size1d; i++)
	{
		for(j=0; j<size2d; j++)
			dup[i][j] = iArrs[i][j];
	}

	return dup;
}
```

inttest.c

```
#include <stdio.h>
#include <stdlib.h>
#include "Integer.h"
#define MAX_SIZE 4

int main(void){

	int* iArr = mallocIntArray(MAX_SIZE);
	int i;

	for(i=0;i<MAX_SIZE;i++)
	{
		iArr[i]=i;
		printf("iArr[%d] = %d\n", i, iArr[i]);
	}

	int* dupArr = duplicateInt(iArr);
	free(iArr);

	for(i=0;i<MAX_SIZE;i++)
	{
		printf("dupArr[%d] = %d\n", i, dupArr[i]);
	}
```

```
	free(dupArr);

	return EXIT_SUCCESS;
}
```

```
~/Documents/courses/cs2263/lecture/lecture11 $ ./test
iArr[0] = 0
iArr[1] = 1
iArr[2] = 2
iArr[3] = 3
dupArr[0] = 0
dupArr[1] = 1
dupArr[2] = 2
dupArr[3] = 3
```

So. I may have assumed what the second part of the fornextday was and you know what that does so I made a deep copy of list of integer arrays. Hope that's somewhat acceptable.

copylistints.c

```
#include"Integer.h"
#include<stdio.h>
#include<stdlib.h>

#define MAX_1D 3
#define MAX_2D 4

int main(void)
{
	int i;
	int *storage = (int*)malloc(MAX_1D*MAX_2D*sizeof(int));
	int **iArrs = (int**)malloc(MAX_1D*sizeof(int*));

	for(i=0; i<MAX_1D; i++)
		iArrs[i] = storage + 4*i;

	int j;
	for(i=0; i<MAX_1D; i++)
	{
		for(j=0; j<MAX_2D; j++)
			iArrs[i][j] = i+j;
	}
```

```
	int** dup = duplicateIntArrays(iArrs, MAX_1D, MAX_2D);
	free(iArrs);
	free(storage);

	for(i=0; i<MAX_1D; i++)
	{
		for(j=0; j<MAX_2D; j++)
			printf("dup[%d][%d] = %d\n", i, j, dup[i][j]);
	}

	free(dup);
	return EXIT_SUCCESS;
}
```