ESc101N: Fundamentals of computing(Lab Session 8)

September 22, 2009

Instructions

1. Please read the question carefully and write the program accordingly
2. Make sure that the TA has graded you program
3. The marks are distributed as follows. You get 60% of the marks if the basic algorithm is current, 20% if you manage to compile and execute and 20% for writing the code cleanly, i.e. using proper variable names, intending and making the code more readable.

Question 1. (a) Write a `C` functions for the following tasks.

i. (2 marks) The function `int ** allocMatrix(int m, int n)` that will allocates, using malloc, space for an $m \times n$ matrix.

ii. (2 marks) The function `void free(int **a, int m)` to free up the memory allocated for an $m \times n$ matrix.

iii. ($\frac{1}{2}$ mark) The function `void readMatrix(int **a, int m, int n)` that reads an $m \times n$ matrix.

iv. ($\frac{1}{2}$ mark) The function `void printMatrix(int **a, int m, int n)` that prints an $m \times n$ matrix.

(b) (5 marks) Write the function `int ** mulMatrix(int **a, int **b, int m, int n, int p)` that returns of an $m \times n$ matrix with an $n \times p$ matrix. Write a program that reads two matrices and multiplies them.

The sample solution is given below

```
Script started on Wed 23 Sep 2009 11:39:23 IST
$ ./a.out
enter the number m of rows of first matrix:3
enter the number n of columns of first matrix/rows of second matrix:2
enter the number p of columns of second matrix:1
enter the first matrix:
        enter [0][0] th entry:2
        enter [0][1] th entry:3
        enter [1][0] th entry:4
        enter [1][1] th entry:2
        enter [2][0] th entry:4
        enter [2][1] th entry:5
enter the second matrix:
        enter [0][0] th entry:2
        enter [1][0] th entry:3
The product of the matrices:
        2       3
        4       2
```

```
        4       5
and
        2
        3
are
        13
        14
        23
$
Script done on Wed 23 Sep 2009 11:39:58 IST
```

Solution:

```
# line 80 "question.lhs"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void readMatrix(int **, int, int);
void printMatrix(int **, int, int);
int **allocMatrix(int, int);
void freeMatrix(int **a, int m);
int ** sumMatrix(int **a, int **b, int m, int n);
int ** mulMatrix(int **a, int **b, int m, int n, int p);

int main(){
  int **a,  **b,  **c;
  int m, n, p;
  printf("enter the number m of rows of first matrix:");
  scanf("%d", &m);
  printf("enter the number n of columns of first matrix/rows of second matrix:");
  scanf("%d", &n);
  printf("enter the number p of columns of second matrix:");
  scanf("%d", &p);
  a = allocMatrix(m, n);
  b = allocMatrix(n, p);
  if( a == NULL || b == NULL)
  {
    printf("not enough memory\n");
    return 1;
  }
  printf("enter the first matrix:\n");
  readMatrix(a,m,n);
  printf("enter the second matrix:\n");
  readMatrix(b,n,p);
  c = mulMatrix(a,b,m,n,p);
  if( c == NULL){
    printf("not enough memory\n");
    return 1;
  }
  printf("The product of the matrices:\n");
```

```
  printMatrix(a,m,n);
  printf("and\n");
  printMatrix(b,n,p);
  printf("are\n");
  printMatrix(c,m,p);

}

int ** sumMatrix(int **a, int ** b, int m, int n)
{
  int **c;
  c = allocMatrix(m,n);
  if( c == NULL) return NULL;
  for(int i = 0; i < m; i ++){
    for(int j = 0; j < n; j ++){
      c[i][j] = a[i][j] + b[i][j];
    }
  }
  return c;
}


void readMatrix(int **a, int m, int n){
  for(int i = 0; i < m; i ++){
    for(int j = 0; j < n; j ++){
      printf("\tenter [%d][%d] th entry:",i,j);
      scanf("%d",&a[i][j]);
    }
  }
}
void printMatrix(int **a, int m, int n){
  for(int i = 0; i < m; i ++){
    printf("\t");
    for(int j = 0; j < n; j++){
      printf("%d\t",a[i][j]);
    }
    printf("\n");
  }
}

int **allocMatrix(int m, int n)
{
  int **a;
  a = (int **) malloc(m * sizeof(int *));
  if( a == NULL) return NULL;
  for(int i = 0 ; i < m; i ++){
    a[i] = (int *) malloc(n * sizeof(int));
    if(a[i] == NULL){
      for(int j = 0; j < i; j ++){
        free(a[j]);
        a[j]=NULL;
```

```
      }
      return NULL;
    }
  }
  return a;
}

int **mulMatrix(int **a, int **b, int m, int n, int p)
{
  int **c;
  c = allocMatrix(m,p);
  if ( c == NULL ) return NULL;
  for(int i = 0; i < m; i ++){
    for(int k=0; k < p; k++){
      c[i][k] = 0;
      for(int j = 0; j < n; j ++){
        c[i][k] = c[i][k]  + a[i][j] * b[j][k];
      }
    }
  }
  return c;
}
```