Name: Swarnabh Paul

Section: Y

Roll no: 19CS8122

Assignment no: 2

Questions attempted: a,b,c,d

**Question (a)**

**Code:**

#include<iostream>

using namespace std;

**void** **insHeap**(**int** \*heap,**int** e,**int** &heap\_size,**int** n);

**void** **delHeap**(**int** \*heap,**int** &heap\_size,**int** n);

**void** **heapify**(**int** \*heap,**int** i,**int** heap\_size);

**int**\* **alloc1DArray**(**int** n);

**void** **display1DArray**(**int** \*heap,**int** heap\_size);

**void** **swap**(**int** &a,**int** &b);

**void** **dealloc1DArray**(**int** \*heap);

**int** **main**()

{

**int** \*heap=NULL;

**int** i, heap\_size=**0**, n, e, choice;

cout<<"Input maximum size of heap: ";

cin>>n;

heap=alloc1DArray(n);

**do**

{

cout<<"Enter choice:**\n**1. Insert element**\n**2. Delete element**\n**3. Display heap**\n**4. Exit**\n**";

cin>>choice;

**switch**(choice)

{

**case** **1**: cout<<"Enter element to be inserted: ";

cin>>e;

insHeap(heap,e,heap\_size,n);

**break**;

**case** **2**: delHeap(heap,heap\_size,n);

**break**;

**case** **3**: display1DArray(heap,heap\_size);

**break**;

**case** **4**: cout<<"Exiting...";

**break**;

}

}**while**(choice!=**4**);

dealloc1DArray(heap);

**return** **0**;

}

**void** **insHeap**(**int** \*heap,**int** e,**int** &heap\_size,**int** n)

{

**if**(heap\_size==n)

{

cout<<"Heap overflow.**\n**";

**return**;

}

**int** i, parent;

heap[heap\_size]=e;

i=heap\_size;

parent=(i-**1**)/**2**;

heap\_size++;

**while**(parent>=**0** && heap[i]>heap[parent])

{

swap(heap[i],heap[parent]);

i=parent;

parent=(i-**1**)/**2**;

}

}

**void** **delHeap**(**int** \*heap,**int** &heap\_size,**int** n)

{

**if**(heap\_size==**0**)

{

cout<<"Heap underflow.**\n**";

**return**;

}

heap[**0**]=heap[heap\_size-**1**];

heap\_size--;

heapify(heap,**0**,heap\_size);

}

**void** **heapify**(**int** \*heap,**int** i,**int** heap\_size)

{

**int** lchild=**2**\*i+**1**, rchild=**2**\*i+**2**, largest=i;

**if**(lchild<heap\_size && heap[lchild]>heap[largest])

largest=lchild;

**if**(rchild<heap\_size && heap[rchild]>heap[largest])

largest=rchild;

**if**(largest!=i)

{

swap(heap[i],heap[largest]);

heapify(heap,largest,heap\_size);

}

}

**int**\* **alloc1DArray**(**int** n)

{

**int** \*t=new **int**[n];

**return** t;

}

**void** **display1DArray**(**int** \*heap,**int** heap\_size)

{

**int** i;

cout<<"Displaying heap as a 1D array as follows:**\n**";

**for**(i=**0**;i<heap\_size;i++)

cout<<heap[i]<<' ';

cout<<endl;

}

**void** **swap**(**int** &a,**int** &b)

{

**int** t=a;

a=b;

b=t;

}

**void** **dealloc1DArray**(**int** \*heap)

{

delete[] heap;

}

**Output:**

Input maximum size of heap: 5

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

1

Enter element to be inserted: 4

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

1

Enter element to be inserted: 10

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

1

Enter element to be inserted: 3

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

1

Enter element to be inserted: 5

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

1

Enter element to be inserted: 1

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

3

Displaying heap as a 1D array as follows:

10 5 3 4 1

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

2

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

3

Displaying heap as a 1D array as follows:

5 4 3 1

Enter choice:

1. Insert element

2. Delete element

3. Display heap

4. Exit

4

Exiting...

**Question (b)**

**Code:**

#include<iostream>

using namespace std;

**int**\*\* **alloc2DArray**(**int** m,**int** n);

**void** **read2DArray**(**int** \*\*m,**int** r,**int** c);

**int**\*\* **multMatrix**(**int** \*\*m1,**int** \*\*m2,**int** r1,**int** c1,**int** r2, **int** c2);

**void** **display2DArray**(**int** \*\*m,**int** r,**int** c);

**void** **dealloc2DArray**(**int** \*\*m,**int** r);

**int**\*\* **cofactor**(**int** \*\*matrix,**int** index,**int** n);

**int** **detMatrix**(**int** \*\*matrix,**int** n);

**int** **main**()

{

**int** m, n, m2, n2, m3, n3;

**char** choice='n';

**int** \*\*matrix=NULL;

cout<<"Enter order of matrix (rows\*cols): ";

cin>>m>>n;

matrix=alloc2DArray(m,n);

read2DArray(matrix,m,n);

cout<<"Want to do matrix multiplication? <y/n>: ";

cin>>choice;

**if**(choice=='y'||choice=='Y')

{

**int** \*\*matrix2=NULL;

**int** \*\*res=NULL;

cout<<"Enter order of second matrix (rows\*cols): ";

cin>>m2>>n2;

**if**(n!=m2)

{

cout<<"Matrix multiplication not possible.";

}

**else**

{

matrix2=alloc2DArray(m2,n2);

read2DArray(matrix2,m2,n2);

m3=m;

n3=n2;

res=multMatrix(matrix,matrix2,m,n,m2,n2);

cout<<"Result is:**\n**";

display2DArray(res,m3,n3);

dealloc2DArray(matrix2,m2);

dealloc2DArray(res,m3);

}

}

**if**(m==n)

{

cout<<"**\n**Determinant of first matrix is: "<<detMatrix(matrix,n);

}

dealloc2DArray(matrix,m);

**return** **0**;

}

**int**\*\* **alloc2DArray**(**int** m,**int** n)

{

**int** \*\*t=new **int**\*[m];

**int** i;

**for**(i=**0**;i<m;i++)

t[i]=new **int**[n];

**return** t;

}

**void** **read2DArray**(**int** \*\*m,**int** r,**int** c)

{

**int** i, j;

cout<<"Input matrix:**\n**";

**for**(i=**0**;i<r;i++)

**for**(j=**0**;j<c;j++)

cin>>m[i][j];

}

**int**\*\* **multMatrix**(**int** \*\*m1,**int** \*\*m2,**int** r1,**int** c1,**int** r2, **int** c2)

{

**int** \*\*r=alloc2DArray(r1,c2);

**int** i, j, k;

**for**(i=**0**;i<r1;i++)

{

**for**(j=**0**;j<c2;j++)

{

r[i][j]=**0**;

**for**(k=**0**;k<c1;k++)

r[i][j]+=(m1[i][k]\*m2[k][j]);

}

}

**return** r;

}

**void** **dealloc2DArray**(**int** \*\*m,**int** r)

{

**int** i;

**for**(i=**0**;i<r;i++)

delete[] m[i];

delete[] m;

}

**void** **display2DArray**(**int** \*\*m,**int** r,**int** c)

{

**int** i, j;

**for**(i=**0**;i<r;i++)

{

**for**(j=**0**;j<c;j++)

cout<<m[i][j]<<' ';

cout<<endl;

}

}

**int** **detMatrix**(**int** \*\*matrix,**int** n)

{

**int** i, det=**0**, f=**1**;

**int** \*\*cofMat=NULL;

**if**(n==**1**)

**return** matrix[**0**][**0**];

**else** **if**(n==**2**)

**return** (matrix[**0**][**0**]\*matrix[**1**][**1**]-matrix[**0**][**1**]\*matrix[**1**][**0**]);

**else**

{

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

{

cofMat=cofactor(matrix,i,n);

det+=(f\*matrix[**0**][i]\*detMatrix(cofMat,n-**1**));

f=-f;

dealloc2DArray(cofMat,n-**1**);

}

**return** det;

}

}

**int**\*\* **cofactor**(**int** \*\*matrix,**int** index,**int** n)

{

**int** \*\*cofMat=alloc2DArray(n-**1**,n-**1**);

**int** i, j, cj=**0**;

**for**(i=**1**;i<n;i++)

{

cj=**0**;

**for**(j=**0**;j<n;j++)

{

**if**(j!=index)

{

cofMat[i-**1**][cj]=matrix[i][j];

cj++;

}

}

}

**return** cofMat;

}

**Output:**

Enter order of matrix (rows\*cols): 3 3

Input matrix:

1 2 3

3 4 5

7 6 4

Want to do matrix multiplication? <y/n>: y

Enter order of second matrix (rows\*cols): 3 3

Input matrix:

5 2 6

5 6 7

7 6 4

Result is:

36 32 32

70 60 66

93 74 100

Determinant of first matrix is: 2

**Question (c)**

**Code:**

#include<iostream>

#include<stdlib.h>

#include<time.h>

#include"myMatrix.h"

using namespace std;

**void** **solve**(**int** \*\*a,**int** \*\*c,**int** n);

**int** \*\***createCramerMatrix**(**int** \*\*a,**int** \*\*c,**int** n,**int** index);

**void** **randomInput2DArray**(**int** \*\*m,**int** r,**int** c);

**int** **main**()

{

**int** n;

**int** \*\*a=NULL;

**int** \*\*c=NULL;

**time\_t** seconds=time(NULL);

srand(seconds);

cout<<"Input no: of variables: ";

cin>>n;

cout<<"The program uses Cramer's to solve a system of linear equations AX=C where matrix A (set of coefficients) and matrix C are taken as inputs.**\n**";

a=alloc2DArray(n,n);

randomInput2DArray(a,n,n);

cout<<"Matrix A after random input:**\n**";

display2DArray(a,n,n);

c=alloc2DArray(n,**1**);

randomInput2DArray(c,n,**1**);

cout<<"Matrix C after random input:**\n**";

display2DArray(c,n,**1**);

solve(a,c,n);

dealloc2DArray(a,n);

dealloc2DArray(c,n);

**return** **0**;

}

**void** **solve**(**int** \*\*a,**int** \*\*c,**int** n)

{

**int** \*\*cram=NULL;

**double** x;

**int** d=detMatrix(a,n), d\_i, i;

**if**(d==**0**)

{

cout<<"This equation cannot be solved.**\n**";

**return**;

}

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

{

cram=createCramerMatrix(a,c,n,i);

d\_i=detMatrix(cram,n);

dealloc2DArray(cram,n);

x=**double**(d\_i)/d;

cout<<"x\_"<<i+**1**<<" = "<<x<<endl;

}

}

**int** \*\***createCramerMatrix**(**int** \*\*a,**int** \*\*c,**int** n,**int** index)

{

**int** i, j;

**int** \*\*cram=alloc2DArray(n,n);

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

{

**for**(j=**0**;j<n;j++)

{

**if**(j==index)

cram[i][j]=c[i][**0**];

**else**

cram[i][j]=a[i][j];

}

}

**return** cram;

}

**void** **randomInput2DArray**(**int** \*\*m,**int** r,**int** c)

{

**int** i, j;

**for**(i=**0**;i<r;i++)

**for**(j=**0**;j<c;j++)

m[i][j]=(rand()%**100**);

}

**Output(1):**

Input no: of variables: 3

The program uses Cramer's to solve a system of linear equations AX=C where matrix A (set of coefficients) and matrix C are taken as inputs.

Matrix A after random input:

92 67 0

24 78 60

45 73 55

Matrix C after random input:

75

74

89

Solution is:

x\_1 = 0.930625

x\_2 = -0.15847

x\_3 = 1.06709

**Output(2):**

Input no: of variables: 10

The program uses Cramer's to solve a system of linear equations AX=C where matrix A (set of coefficients) and matrix C are taken as inputs.

Matrix A after random input:

0 17 94 81 79 70 52 77 81 97

93 42 72 6 15 31 10 99 47 1

31 66 81 4 86 61 28 94 14 43

56 82 39 14 13 59 70 57 47 86

85 72 70 46 80 62 2 46 71 42

83 20 29 61 9 12 24 63 36 71

38 4 82 45 87 73 33 37 36 3

14 55 96 51 66 74 5 75 69 55

65 72 64 2 17 60 51 52 2 66

18 69 38 19 19 26 73 35 25 53

Matrix C after random input:

85

73

20

16

82

79

69

88

63

66

Solution is:

x\_1 = 0.451891

x\_2 = -0.499757

x\_3 = 0.392488

x\_4 = 1.66165

x\_5 = -1.14214

x\_6 = 1.52134

x\_7 = -1.51942

x\_8 = -1.32621

x\_9 = -1.51785

x\_10 = -0.650837

**Question (d)**

**Code:**

#include<iostream>

#include<time.h>

using namespace std;

**const** **int** n=**6**;

**unsigned** **long** **int** **myrand**(**unsigned** **long** **int**);

**void** **displayMarks**(**int** \*marks[n],**int** num[n]);

**int** **findDeptTopper**(**int** \*marks,**int** num);

**void** **findBatchTopper**(**int** \*marks[n],**int** num[n]);

**int** **main**()

{

**time\_t** seconds;

seconds=time(NULL);

**unsigned** **long** **int** seed=seconds;

**int** \*marks[n];

**int** i, noOfStud[n], j;

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

{

cout<<"How many students in department "<<i+**1**<<": ";

cin>>noOfStud[i];

marks[i]=new **int**[noOfStud[i]];

}

cout<<"Entering marks of students roll no. wise in each department.";

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

{

**for**(j=**0**;j<noOfStud[i];j++)

{

seed=myrand(seed);

marks[i][j]=(seed%**100**)+**1**;

}

}

displayMarks(marks,noOfStud);

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

{

**int** top=findDeptTopper(marks[i],noOfStud[i]);

cout<<"Topper of department "<<i+**1**<<" is roll no. "<<top<<" scored "<<marks[i][top-**1**]<<" marks."<<endl;

}

findBatchTopper(marks,noOfStud);

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

delete[] marks[i];

**return** **0**;

}

**unsigned** **long** **int** **myrand**(**unsigned** **long** **int** x)

{

**unsigned** **long** **long** **int** m=**2147483647**, a=**65539**;

**unsigned** **long** **int** r=(x\*a)%m;

**return** r;

}

**void** **displayMarks**(**int** \*marks[n],**int** num[n])

{

**int** i, j;

cout<<"**\n**Displaying marks roll no. wise for each department.**\n**";

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

{

cout<<"Department "<<i+**1**<<": ";

**for**(j=**0**;j<num[i];j++)

cout<<marks[i][j]<<' ';

cout<<endl;

}

}

**int** **findDeptTopper**(**int** \*marks,**int** num)

{

**int** i, m=**0**;

**for**(i=**1**;i<num;i++)

**if**(marks[i]>marks[m])

m=i;

**return** m+**1**;

}

**void** **findBatchTopper**(**int** \*marks[n],**int** num[**6**])

{

**int** topDept=**0**, top=**0**, i, j;

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

{

**for**(j=**0**;j<num[i];j++)

{

**if**(marks[i][j]>marks[topDept][top])

{

topDept=i;

top=j;

}

}

}

cout<<"Batch topper in department "<<topDept+**1**<<", has roll no. "<<top+**1**<<" and scored "<<marks[topDept][top]<<" marks.";

}

**Output:**

How many students in department 1: 10

How many students in department 2: 9

How many students in department 3: 8

How many students in department 4: 12

How many students in department 5: 15

How many students in department 6: 6

Entering marks of students roll no. wise in each department.

Displaying marks roll no. wise for each department.

Department 1: 49 74 50 25 2 39 69 84 70 75

Department 2: 15 73 29 53 11 3 90 9 79

Department 3: 7 91 46 88 2 45 46 16

Department 4: 70 57 1 9 55 4 94 32 83 64 58 49

Department 5: 90 46 95 56 67 53 98 66 47 81 53 100 31 76 88

Department 6: 92 97 81 56 24 1

Topper of department 1 is roll no. 8 scored 84 marks.

Topper of department 2 is roll no. 7 scored 90 marks.

Topper of department 3 is roll no. 2 scored 91 marks.

Topper of department 4 is roll no. 7 scored 94 marks.

Topper of department 5 is roll no. 12 scored 100 marks.

Topper of department 6 is roll no. 2 scored 97 marks.

Batch topper in department 5, has roll no. 12 and scored 100 marks.