תאריך: **23.11.99**

שם הקורס: **שפות תוכנה**

משך הבוחן: **שעה וחצי**

חומר עזר: **2 דפי פוליו כתובים**

- בכל תוכנית חייבים להיות הסברים, ושמות המשתנים חייבים להיות בעלי משמעות.

1. נתונה תוכנית ללא הסברים ועם חלקים חסרים כגון: חלק מה-main, חלק מהפונקציות והאופרטורים. אליכם להבין את התוכנית, להסביר את החלקים אשר ידרשו ולהשלים את הפרטים החסרים. באופן כלל ישנו Class Cell אשר מכיל מצביעה לוקטור. ישנה אופציה למדוד מרחק בין וקטור זה לוקטור חיצוני, וכן לעדכן את הוקטור על סמך הוקטור החיצוני (ראה נוסחת מרחק ונוסחת עדכון בסוף המבחן).

1.1 הסבר את שלבי התוכנית הראשית.

1.2 השלם את התוכנית הראשית במקום ההערות בעברית.

1.3 הסבר מה עושה ה-constructor.

1.4 הנח שאסור להוסיף פונקציות נוספות ב-class Cell פרט ל-constructors, destructors ו-operators. השלם את הדברים הנחוצים. זכור class members מותר להוסיף רק ב-private.

1.5 כתוב את ה-member functions: TrainCell ו-Distance.

- **נוסחאות:**

מרחק: $Distance = (m\_pdCell - Vector) * (m\_pdCell - Vector)$

לימוד: $m\_pdCell = m\_pdCell - (m\_pdCell - Vector) * LearningRate$

שימו לב – ה-LearningRate הוא משתנה שערכו זהה אצל כל האוביקטים ובסוף כל הפעלה של TrainCell ערכו ישתנה: $LearningRate = 0.9 \cdot LearningRate$

ערכו ההתחלתי הוא: $LearningRate = 0.99$

בהצלחה

```
#include <stdlib.h>
#include <iostream.h>
#include "Cell.h"

int main ( void )
{
  unsigned short   usCellsNumber                  ;
  unsigned short   usCellsSize                    ;
  unsigned short   usInputVectorsNumber           ;
  double      ** ppdData                          ;
  Cell      * pCells                              ;

  cout << "Input the Numper of Cells: "           ;
  cin >> usCellsNumber                            ;
  cout << "Input Cells Size: "                    ;
  cin >> usCellsSize                              ;
  cout << "How Many Input Vectors Do You Have? "  ;
  cin >> usInputVectorsNumber                     ;

  if ( usInputVectorsNumber > 0 )
  {
    unsigned shortusCounter =0                    ;
    unsigned shortusTempSize                      ;

    usTempSize = usInputVectorsNumber *
            usCellsSize              ;
    ppdData = new double * [ usInputVectorsNumber ] ;
    ppdData [ 0 ] = new double [ usTempSize ]     ;

    for ( ; usCounter < usTempSize ; usCounter ++ )
    {
      cout << "Input a New Value: "               ;
      cin >> ppdData [ 0 ] [ usCounter ]          ;
    }

    for ( usCounter = 1 ;
          usCounter < usInputVectorsNumber ;
          usCounter ++ )
    {
      ppdData [ usCounter ] =
                & ppdData [ usCounter - 1 ]
                [ usCellsSize ]                   ;
    }
```

אתחל את pCell בצורה נאותה. – כלומר לאתחל את כול התאים כמו שמתואר ב-Constructor.

```
    for ( usCounter = 0 ;
        usCounter < usInputVectorsNumber ;
        usCounter ++ )
    {
      double        dMinDistance                  ;
      unsigned short   usMinArgument = 0          ;
      unsigned short   usInCounter = 1            ;

      dMinDistance = pCells [ 0 ] . Distance ( ppdData [ usCounter ] )
  ;

      for ( ; usInCounter < usCellsNumber ;
           usInCounter ++ )
      {
        double      dTmpDist                      ;
```

```
          dTmpDist = pCells [ usInCounter ] . Distance ( ppdData [
usCounter ] ) ;

          if ( dTmpDist < dMinDistance )
          {
            dMinDistance = dTmpDist        ;
            usMinArgument = usInCounter      ;
          }
        }

        pCells [ usMinArgument ] . TrainCell ( ppdData [ usCounter ] )  ;

      }

   }

   return 0  ;
}


/*********************************************************************/

#ifndef   SE_MID_TEST_SEMESTER1_1999_A
#define   SE_MID_TEST_SEMESTER1_1999_A

#include <stdlib.h>
#include <iostream.h>
#include <assert.h>
#include <memory.h>


class Cell
{
public :
   Cell ( const unsigned short    usCellsSize = 0 )
   {
     m_usCellsSize = usCellsSize                  ;
     HelpInit ( )                           ;
   }
   void InitCell ( const unsigned short usCellsSize )
   {
     if ( m_usCellsSize != 0 )
     {
       delete [ ] m_pdCell                      ;
     }

     m_usCellsSize = usCellsSize                  ;
     HelpInit ( )                           ;
   }

   double Distance ( const double * const pdInVector )    ;
   void TrainCell ( const double * const pdInVector )     ;
```

השלם אופרטורים constructors ו-destructor.

```
private :
```

השלם משתנים הכרחיים.

```
   double           *  m_pdCell              ;
   static double        s_dLearningRate           ;
```

```
  void             HelpInit ( )            ;
}                                      ;

#endif


/*********************************************************************/

#include "Cell.h"

double Cell :: Distance (const double * const pdInVector )
{
```

השלם את גוף הפונקציה.

```
}

void Cell :: TrainCell ( const double * const pdInVector )
{
```

השלם את גוף הפונקציה.

```
}

void Cell :: HelpInit ( )
{
  if ( m_usCellsSize )
  {
    unsigned short   usCounter = 0             ;

    m_pdCell = new double [ m_usCellsSize ]           ;
    for ( ; usCounter < m_usCellsSize ; usCounter ++ )
    {
      m_pdCell [ usCounter ] *= ( double ) rand ( ) /
                         RAND_MAX - 0.5      ;
    }
  }
}
```