Objective:

- To give you quick review of debugging, memory issues, pointers, streams, and logic ☺.
- To give you review of templates, array of object, and logic ☺.

Note:

If you feel any difficulties in implementing these tasks then feel free to discuss with me or T.As but not with your peers. If you don't know the concept of watch in debugging then ask from T.As.

Lab Part-A

Focuses on debugging, memory issues, pointers, streams, and logic.

Question 1: **(2)**

Declare an alias named as 'ref' of identifier 'p' in the following code.

```
const int * q = new int;
const int *const * p = & q;
const int * const * & ref = p;
```

Question 2: **(3)**

Consider the declaration 'int * const * * const p;'. Circle the expression(s) below which can be treated as an L-Value.

A. * p **B.** * * p **C.** * * * p **D.** & p **E.** p

Question 3: **(1)**

What does the following declaration mean?

```
int (*ptr)[10];
```

A. ptr is a pointer to a row size 10
B. ptr is an array of size 10 whose each location is of type pointer to integer.
C. It is a syntax error to declare ptr like this.
D. None of the above

Question 4: **(2)**

Consider the code below:

```
void f()
{
        int x = 23;
        int *p = new int;
        p = new int[50];
        p = & x;
}
```

```
void main()
{
        f();
}
```

Circle the type of errors the above code may produce.

A. Null pointer **B.** Dangling pointer **C.** Illegal memory access
D. Memory leakage **E.** None of the above

Question 5: **(4)**

What will be displayed on console when the following code is executed?

```
struct Set
{
        int *data;
        int noOfElements;
        int capacity;
};
```

```
void main()
{
        Set t;
        t.noOfElements = 0;
        t.capacity = 10;
        t.data = new int [ t.capacity ];
        for ( int i=0; i < t.capacity; i=i+1)
        {
                t.data[i] = i;
        }
        int * p = (int*) &t;
        p[1] = 15;
        p[2] = 30;
        cout << t.data[1] << endl << t.noOfElements << endl
             << t.capacity << endl;
}
```

OUTPUT

1

15

30

Question 6: **(2)**

Shallow copy for the objects of 'Set' struct given in Question No. 5 will be safe?

A. True **B.** False

Question 7: **(2)**

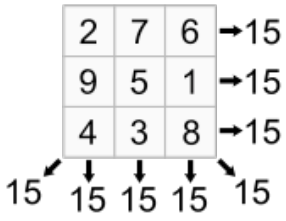
Use 'memcpy' function to copy the contents of following array 'a' into array 'b'.

```
int a[5]= {1, 2, 3, 4, 5};
int b[5]= {14, 12, 5, 14, 21};
memcpy(b, a, sizeof(b));
```

Question 8: **(10)**

A magic square is a square matrix in which elements are arranged in such a way that the numbers in each row, and in each column, and the numbers in the forward and backward main diagonals, all add up to the same number.

Following is an example of Magic Square of order 3: In each row/column/diagonal the sum is same.



Consider a text file named "abc.txt" which contains a square matrix in it.

The file format is as follows:

- First line contains the order of matrix

- Each subsequent line contains row of matrix with each element separated by space.

A sample file 'abc.txt'

```
3
2  7  6
9  5  1
4  3  8
```

Write a function, which reads the matrix from the file and return true if the stored matrix is a magic square otherwise returns false.

Function Prototype:

bool isMagicSquare ( )

```
bool isMagicSquare()
{
    fstream fs("abc.txt",ios::in);

    int N;
    fs >> N;
    int * * mat = new int * [N];
    for ( int i=0; i<N; i++)
    {
        mat[i] = new int[N];
    }
    for ( int i=0; i<N; i++)
    {
        for ( int j=0; j<N; j++)
        {
            fs >> mat[i][j];
        }
    }
    int istRowSum=0;
    for ( int i=0; i<N; i++)
    {
        istRowSum = istRowSum + mat[0][i];
    }
    int rowSum=0;
    for ( int i=1; i<N; i++)
    {
        rowSum=0;
        for ( int j=0; j<N; j++)
        {
            rowSum = rowSum + mat[i][j];
        }
        if (rowSum != istRowSum)
            return false;
    }
    int colSum = 0;
    for ( int i=1; i<N; i++)
    {
        colSum=0;
        for ( int j=0; j<N; j++)
        {
            colSum = colSum + mat[j][i];
        }
        if (colSum != istRowSum)
            return false;
    }
    return true;
}
```

You can study in detail about magic square at http://en.wikipedia.org/wiki/Magic_square

Question 9: **(10)**

Write a function named *'stringTokenizer'*, which returns a substring by extracting it from the given string depending upon on the delimiter characters passed.

Sample Run:

```
void main()
{
        char str[30] = "- This, a sample string.";
        char * token = stringTokenizer ( str, ",.-");

        // After the execution of above statement:
        // the token points to " This"
        // the str contents becomes " a sample string."
        // So if we call it again

        token = stringTokenizer ( str, ",.-");

        // then token points to "a"
        // str contents becomes "sample string."
}
```

Considering the above sample run: the 'stringTokenizer' search for a substring in 'str' which starts from any character given in delimiter string i.e. " ,.-" and ends at any delimiter character and returns the substring.

Function Prototype:

*char * stringTokenizer (char * str , const char * delim);*

```
char * stringTokenizer( char * str, char * del)
{
    int delStart = -1;
    int i=0,j=0;
    //find delimiter start
    while( str[i]!='\0' && delStart == -1)
    {
        j=0;
        while( del[j]!='\0' && delStart == -1)
        {
            if (str[i]==del[j])
            {
                delStart = i;
            }
            j++;
        }
        i++;
    }
    if (delStart==-1)
        return 0;
    int delEnd = -1;
    i=delStart+1,j=0;

    //find delimiter end
    while( str[i]!='\0' && delEnd == -1)
    {
        j=0;
        while( del[j]!='\0' && delEnd == -1)
        {
            if (str[i]==del[j])
            {
                delEnd = i;
            }
            j++;
        }
        i++;
    }
    if(delEnd==-1)
    {
        return 0;
    }
    int tokenLength = delEnd - delStart;
    char * token = new char[ tokenLength + 1 ];
    i=delStart+1;
    j=0;
    //copy token
    while(i<delEnd)
    {
        token[j] = str[i];
        i++;j++;
    }
    token[tokenLength]='\0';

    i=0;
    j=delEnd+1;

    //remove token from str
    while (str[j]!='\0')
    {
        str[i]=str[j];
        i++;j++;
    }
    str[i]='\0';

    return token;
}
int main()
{
    char str[30] = "- This, a sample string.";
    char * token = stringTokenizer ( str, " ");

    // After the execution of above statement:
    // the token points to "This,"
    // the str contents becomes "a sample string."
    // So if we call it again

    token = stringTokenizer ( str, " ");

    // then token points to "a"
    // str contents becomes "string."
    return 1;
}
```


Lab Part-B

Focuses on of templates, array of object, and logic ☺.

A polynomial is an equation of the form:

$P(x) = a_0x^{e0} + a_1x^{e1} + \ldots + a_nx^{en}$ where a_i belongs to Coefficient e_i belongs to Exponent.

Your target is to store this form of equation in Computer and be able to perform the following operations on it.

Selection of data members for this ADT is upto you but you should be able to justify the need of all data members. So, before you start coding, discuss and give your justification to the instructor/T.A for selecting the particular data members and prototypes of the required methods.

You should always implement the Def-Ctor/Copy-Ctor/Dtor/= methods in case of dynamic memory allocation to class data members.

Public Functions:

1. **Constructor**
2. **addTerm (coefficient, power)**
   Add a new term x^{power} term in the polynomial and sets its coefficient.
3. **getDegree ()**
   Returns the degree of a polynomial. For example, when p1 = $4x^5 + 7x^3 - x^2 + 9$, p1.degree () is 5.
4. **getCoefficient (power)**
   Returns the coefficient of the x^{power} term. For example, p1.getCoefficient (3) is 7
5. **operator (value)**
   For example, given a polynomial in "p1", p1(x) will evaluate the polynomial for the given value of x.
6. **Dtor/Copy-Ctor/=**
7. **Arithmetic Operation +**
   Add two polynomials. For instance, assuming p1 = $4x^5 + 7x^3 - x^2 + 9$ and p2 = $6x^4 + 3x^2 + 2x$, then the answer of p1 + p2 must be $4x^5 + 6x^4 + 7x^3 + 2x^2 + 2x + 9$.

→ Try the following methods at home

8. **derivative ()**
   Return a polynomial that is the derivative of the given polynomial. For instance, assuming p1 = $4x^5 + 7x^3 - x^2 + 9$, the derivative polynomial would be $20x^4 + 21x^2 - 2x$.
9. **antiDerivative ()**
   Return a polynomial that is the anti-derivative of the given polynomial. For instance, assuming p1 = $20x^4 + 21x^2 - 2x$, the derivative polynomial would be $4x^5 + 7x^3 - x^2 + C$, Where C is a random constant value.
10. **addToCoefficient (coefficient, power)**
    Adds the given amount to the coefficient of the x^{power} term. For example, p1.addToCoefficient (2.5, 3) changes the coefficient of x^3 to 9.5.
11. **clear ()**
    Set the coefficient of all terms in the polynomial to zero.
12. **setCoefficient (newCoefficient, power)**
    Sets the coefficient of the x^{power} term with newCoeffcient. For instance, p1.setCoefficient (-3, 7) produces the polynomial p1 = $-3x^7 + 4x^5 + 7x^3 - x^2 + 9$. Note that if the term does not exist in the polynomial, it is added.
13. **Arithmetic operations *, -**

Success isn't a result of spontaneous combustion. You must set yourself on fire.
...Arnold H. Glasow...