National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Computer Programming	Course Code:	CS-103
	Program:	BS(CS)	Semester:	Spring 2018
	Duration:	3 Hours	Total Marks:	80
	Paper Date:	21-5-2018	Weight	40
	Section:	ALL	Page(s):	4
	Exam Type:	Final	Roll-Number - Section	

Instruction/Notes: Please solve the exam on the answer sheet. The order of questions should remain same on answer sheet as on question paper.

Question 1: [5*4=20 Marks]

Write the output for the code snippets given below:

Part (a)

```
class A{
private:
  int a;
public:
  A(int x = 10){
       a = x;
       cout<<"A() Called.\n";
  }
  ~A(){
    cout<<"~A()Called for "<<a<<" .\n";
  }
  void Function(A b){
    cout<<"--------------\n";
  }
};
   -------------------------------------
void main(){
       A a;
       A* aptr = &a;
       aptr = new A(5);
       a.Function(*aptr);
       delete aptr;
}
```

Part (b)

```
class A{
       static int * ptr;
       int data;
public:
       A(){
         data = 10;
         ptr = 0;
       }
       static void Print(){
         if(ptr != 0)
            cout<<"*ptr = "<<*ptr<<endl;
       }
};

int* A::ptr = new int(10);
---------------------------------------
void main(){
       A::Print();
       A obj2;
       A::Print();
}
```

Part (c)

```
class XYZ
{
private:
  int x;
public:
  XYZ(int y = 10){
    x = y;
    cout<<"XYZ() called for "<<x<<endl;
  }
  void Print(){
    cout<<x<<endl;
  }
  ~XYZ()      {
    cout<<"~XYZ() Called.\n";
  }
};
    ----------------------------------------
void main()
{
       ABC* x = new ABC;
       x->Print();
       XYZ* ptr = &(x->a);
       delete x;
       ptr->Print();
}
```

```
class ABC
{
  int c;
public:
  XYZ a;
  XYZ* b;
  ABC(int val = 50){
    c = val;
    cout<<"ABC() called for "<<c<<endl;
    b = new XYZ(a);
  }
  void Print(){
    cout<<"c = "<<c<<endl;
    cout<<"a = ";a.Print();
    if(b != nullptr)
    {

       cout<<"b =  ";

       b->Print();

    }
  }
};
```

Part (d)

```
class Vehicle {
public:
	Vehicle() {
		cout<<"Vehicle() called.\n";
	}
	~Vehicle() {
		cout<<"~Vehicle() called.\n";
	}
	virtual void Print()	{
		cout<<"Test\n";
	}
};
---------------------------------------------
class MotorCycle: public Vehicle
{
public:
	MotorCycle() {
		cout<<"MotorCycle() called.\n";
	}
	~MotorCycle(){
		cout<<"~MotorCycle() called.\n";
	}
};
```

```
class Car: public Vehicle{
public:
	Car()  {
		cout<<"Car() called.\n";
	}
	~Car() {
		cout<<"~Car() called.\n";
	}
	virtual void Print()	{
		cout<<"Check\n";
	}
};
---------------------------------------------
void main(){
	Vehicle* vehicles[3];
	vehicles[0] = new MotorCycle;
	vehicles[1] = new Car;
	vehicles[2] = new Vehicle;
	for(int i = 0 ; i<3 ; i++)
		vehicles[i]->Print();
	for(int i = 0 ; i<3 ; i++)
		delete vehicles[i];
}
```

Solution:

(a)

```
A() Called.
A() Called.
--------------
~A()Called for 5 .
~A()Called for 5 .
~A()Called for 10 .
```

(b)

```
*ptr = 10
```

(c)

```
ABC() called for 50
c = 50
a = 10
b =  10
~XYZ() Called.
10
```

(d)

```
Test
Check
Test
~Vehicle() called.
~Vehicle() called.
~Vehicle() called.
```

Question 2: **[20 Marks]**

Dr. Sprinkler, a grammar expert, has come up with an algorithm to correct the placement of commas in an English language text. It's called Sprinkler's Algorithm and is described below.

Sprinkler's Algorithm
Step 1. If a word anywhere in the text is preceded by a comma (i.e. the comma comes before the word), find all occurrences of that word in the text and put a comma before each of those occurrences, except in the case where such an occurrence is the first word of a sentence or already preceded by a comma. **Step 2.** If a word anywhere in the text is succeeded by a comma, (i.e. the comma comes after the word) find all occurrences of that word in the text, and put a comma after each of those occurrences, except in the case where such an occurrence is the last word of a sentence or already succeeded by a comma. **Step 3.** Apply Steps 1 and 2 repeatedly until no new commas can be added to the text.

Note that the text must always remain in the standard format, which follows these rules:

- The text begins with a word.
- Between every two words in the text, there is either a single space, a comma followed by a space, or a period followed by a space (denoting the end of a sentence and the beginning of a new one).
- The last word of the text is followed by a period with no trailing space.
- The first letter of each sentence is capitalized.

Example
Consider the text: **Please sit spot. Sit spot, sit. Spot here now here.**
(note that this text is in the standard format)

Step 1 produces: Please, sit spot. Sit spot, sit. Spot here now here.
Step 2 produces: Please, sit spot. Sit spot, sit. Spot, here now here.
Step 1 produces: Please, sit spot. Sit spot, sit. Spot, here now, here.
STOP, as no more commas can be added to text.

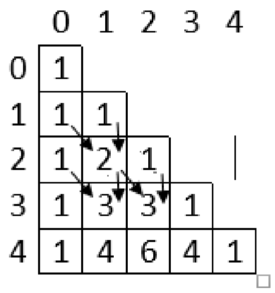
Implement Sprinkler's Algorithm in a function called **SprinkleCommas**. The function accepts a dynamic character array of exactly the same size as the text. Your function updates this same array and makes sure that at the end the size of the array is still exactly the same as the size of the text.

Solution:

```
void sprinklersAlgo(char*& text){
     while(precedingCheck(text) ||
               succeedingCheck(text) );
}

bool precedingCheck(char*& text){
     bool flag = false;
     int wordsSize;
     char* words [];
     tokenize(text,words,wordsSize); // get all words

     for(int i=0; i < wordsSize; i++){
          if(precedesComma(text,words[i])){ // comma precedes word in text
               for (int j=0; i != j && j < wordsSize; j++){
                    if (equals(words[i],words[j])){
                         flag = prependComma(words[j]); // insert if required
                    }
               }
          }
     }

     text = unify(words); // concatenate all words according to given rules
     return flag;
}

bool succeedingCheck(char*& text){
     bool flag = false;
     int wordsSize;
     char* words [];
     tokenize(text,words,wordsSize); // get all words

     for(int i=0; i < wordsSize; i++){
          if(succeedsComma(text,words[i])){ // comma succeeds word in text
               for (int j=0; i != j && j < wordsSize; j++){
                    if (equals(words[i],words[j])){
                         flag = appendComma(words[j]); // insert if required
                    }
               }
          }
     }

     text = unify(words); // concatenate all words according to given rules
     return flag;
}
```

Question 3: **[5+5+10 = 20 Marks]**

In the **Pascal's triangle** of numbers (shown on the right) each row is computed in the following manner:



- In the 0th row, there is a unique nonzero entry 1.
- In subsequent rows the first and last entries are 1.
- The remaining entries of the ith row (where i>0) are computed by adding each consecutive pair of numbers in the (i-1)th row. This is shown by the arrows in the picture on the right

Here's a class to store a Pascal's triangle with n rows.

```
class PascalTriangle{
      int ** triangle; //to store the numbers in the triangle
      int n; //total number of rows

      public:
            PascalTriangle(int nrows){
            //implement this function
            }

            ~PascalTriangle(){
            //implement this function
            }
      };
```

(a) Implement the constructor PascalTriangle(int nrows) to compute the triangle for n rows. For example, for a triangle such as in the picture above n=5, meaning it has rows numbered from 0 to 4. Make sure that the triangle occupies exactly the amount of space needed to store all the numbers.

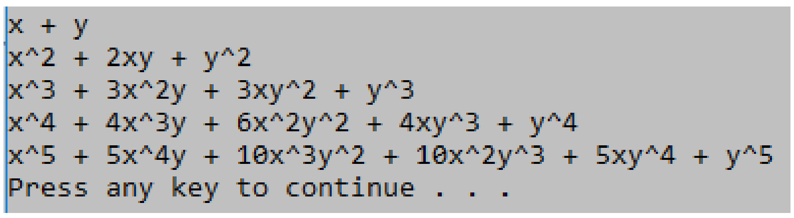
(b) Implement the destructor ~PascalTriangle() to deallocate the triangle.

Now please examine the following polynomials:

$$(x+y)^0 = 1$$
$$(x+y)^1 = x+y$$
$$(x+y)^2 = x^2 + 2xy + y^2$$
$$(x+y)^3 = x^3 + 3x^2 y + 3x y^2 + y^3$$
$$(x+y)^4 = x^4 + 4x^3 y + 6x^2 y^2 + 4x y^3 + y^4$$

As you might have noticed, the coefficients of the binomial **$(x+y)^k$** are exactly the numbers in the row number **k** of the Pascal's triangle! This is a very useful application of the Pascal's Triangle. If we wish to print the polynomial that results from the expansion of **$(x+y)^k$** we simply look at row number **k** of the Pascal's triangle. Of course, we will also need to compute the exponents of x and y in the expansion.

(c) Write a global method `void printBinomialExpansion(int k)` which prints the expansion of **$(x+y)^k$** in the format shown below. This method may use the class PascalTriangle for this purpose. Note: this is a global method. Make sure that it can access the triangle inside the a PascalTriangle object.



```
x + y
x^2 + 2xy + y^2
x^3 + 3x^2y + 3xy^2 + y^3
x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4
x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5
Press any key to continue . . .
```

Note: There should be no memory leaks or dangling pointers in your code.

Solution:

```
class PascalTriangle{
     friend void printBinomialExpansion(int k);

     int ** triangle; //to store the numbers in the triangle
     int n; //total number of rows

public:
     PascalTriangle(int nrows){
        n = nrows;
          triangle = new int*[n];

          for(int i=0; i < n; i++){
               triangle[i] = new int[i+1];
               triangle[i][0] = 1;
               triangle[i][i] = 1;

               for(int j=1; j < i; j++){
                    triangle[i][j] = triangle[i-1][j-1] + triangle[i-1][j];
               }
          }
     }

     ~PascalTriangle(){
          for(int i=0; i < n; i++){
               delete [] triangle[i];
          }
          delete [] triangle;
     }
};

void printBinomialExpansion(int k){

     PascalTriangle pt(k+1);

     if(k == 1){
       cout << "x";
     }
     else{
       cout << "x^" << k;
     }

     cout << "+";

     for(int i=1; i < k; i++){
          cout << pt.triangle[k][i];
          if((k - i) == 1){
               cout << "x";
          }
          else{
               cout << "x^" << k-i;
          }

          if(i == 1){
               cout << "y" << "+";
          }
          else{
               cout << "y^" << i << "+";
          }
     }

     if(k == 1){
       cout << "y";
     }
```

```
      else{
        cout << "y^" << k;
      }

      cout << endl;
}
```

Question 4: **[20 Marks]**

Consider the case of a software that can assist the instructors in preparing exams. An exam can have multiple questions where each question is a multiple-choice, fill-in-blank, or any other suitable subtype. An instructor can add questions to exam and finally print it. Your task is to write an object-oriented program using concepts of associations, inheritance and polymorphism, while satisfying the following description of classes:

- **Exam** can have multiple questions, with the option to **add** questions and **print** the exam.
- **Question** is an abstract (pure virtual) class
- **MCQ** (multiple-choice) is a **Question** containing **statement** and multiple numbered **options**
- **FIB** (fill-in-blank) is a **Question** containing **statement** with a specific **word** left as blank
- The given code of the **main** and corresponding output serves as a guideline for implementation. Take care of memory management issues in your implementation.

Code	`Exam final(2); // exam has 2 questions`
	
	`char* options[] = {"option 1","option 2","option 3"}; // options for mcq question`
	
	`// adding MCQ as q1 having 3 options`
	`final.add(new MCQ("Select the correct option below:",3,options));`
	`// adding FIB as q2 where 4th word is a blank`
	`final.add(new FIB("A fill in blank question",4));`
	
	`final.print(); // prints the exam questions polymorphically`

Output	Question 1 Select the correct option below: 1. option 1 2. option 2 3. option 3 Question 2 A fill in _____ question

Hint: Consider the function string tokenizer function **char*[] strtok(char* text, char* sep)** is already available to you for use. It returns an array of words in the given **text** separated by the separator **sep**.

Solution:

```
class Exam {
protected:
      int questionsNumber;
      int last;
      Question** questions;
public:
      Exam();
      Exam(int);
      void addQuestion(Question *);
      virtual void print();
      virtual ~Exam();
};

Exam::Exam() {
      questionsNumber = 0;
      last = -1;
      questions = 0;
}

Exam::Exam(int n){
      questionsNumber = n;
      questions = new Question*[questionsNumber];
      last = -1;
}

void Exam::addQuestion(Question *q){
      if(++last < questionsNumber){
            questions[last] = q;
      }
}

void Exam::print(){
      for(int i=0; i <= last; i++ ){
            cout << "Question " << i+1 << endl;
            questions[i]->print();
            cout << endl;
      }
}

Exam::~Exam() {
      if(questions != 0){
            for(int i=0; i <= last; i++){
                  delete questions[i];
            }

            delete [] questions;
      }
}

class Question {
public:
      virtual void print() = 0;
};

class MCQ: public Question {
protected:
      char* text;
      int optionsNumber;
      char** options;
public:
      MCQ();
      MCQ(char* t, int n, char* o[]);
      virtual void print();
      virtual ~MCQ();
};


MCQ::MCQ() {
      text = NULL;
      optionsNumber = 0;
      options = NULL;
}

MCQ::MCQ(char* t,int n,char* o[]){
      text = new char[strlen(t)+1];
      strcpy(text,t);

      optionsNumber = n;
      options = new char*[n];
      for (int i=0; i < n; i++){
            options[i] = new char[strlen(o[i])+1];
            strcpy(options[i],o[i]);
      }
}
```

```
void MCQ::print(){
      cout << text << endl;
      for (int i=0; i < optionsNumber; i++){
            cout << "  " << i+1 << ". " << options[i] << endl;
      }
}

MCQ::~MCQ() {
      if (text != NULL){
            delete [] text;
      }

      if(options != NULL){
            for(int i=0; i < optionsNumber; i++){
                  delete [] options[i];
            }
      }

      delete [] options;
}

class FIB: public Question {
protected:
      char* text;
      int blank;

public:
      FIB();
      FIB(char* t,int b);
      virtual void print();
      virtual ~FIB();
};

FIB::FIB() {
      text = NULL;
      blank = -1;
}

FIB::FIB(char* t,int b){
      text = new char[strlen(t)+1];
      strcpy(text,t);
      blank = b;
}

void FIB::print(){

      char* word = strtok(text," ");
      int count = 1;
      while (word != NULL){
            if(count == blank){
                  for(int i=0; i < strlen(word); i++) cout << '_';
            }
            else {
                  cout << word;
            }
            cout << " ";
            count++;
            word = strtok(NULL," ");
      }
      cout << endl;

}

FIB::~FIB() {
      if (text != NULL){
```

```
                delete [] text;
        }
}
```