# 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)** | **Part (b)** |
|---|---|

**Part (a)**
```cpp
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)**
```cpp
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)**

```cpp
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();
}
```

```cpp
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();
            }
      }
};
```

```
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];
}
```

## 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.

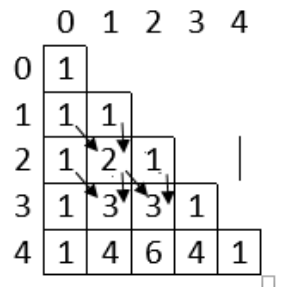| **Example** |
| --- |
| Consider the text: **Please sit spot. Sit spot, sit. Spot here now here.** |
| (note that this text is in the standard format) |
| |
| Step 1 produces:  Please, sit spot. Sit spot, sit. Spot here now here. |
| Step 2 produces:  Please, sit spot. Sit spot, sit. Spot, here now 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.

**Department of Computer Science**

# 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 $0^{th}$ row, there is a unique nonzero entry 1.
- In subsequent rows the first and last entries are 1.
- The remaining entries of the $i^{th}$ 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

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 |   |   |   |   |
| 1 | 1 | 1 |   |   |   |
| 2 | 1 | 2 | 1 |   |   |
| 3 | 1 | 3 | 3 | 1 |   |
| 4 | 1 | 4 | 6 | 4 | 1 |

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

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.

# 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** | ```cpp
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<br>Select the correct option below:<br> 1. option 1<br> 2. option 2<br> 3. option 3<br><br>Question 2<br>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**.