National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Object-oriented Programming**	**Course Code:**	**CS-217**
Program:	**BS (Computer Science)**	**Semester:**	**Spring 2020**
Duration:	**90 Minutes**	**Total Marks:**	**35**
Paper Date:	**'19-Oct-20**	**Weight**	**12.5 %**
Section:	**ALL**	**Page(s):**	**6**
Exam:	**Midterm-I**	**Reg. No.**	

Instruction/Notes: Please solve the exam on paper. No answer sheets to be attached.

Question 1 **(5+5=10 points)**

For code segments given below identify **output or error**. In case of error **highlight the line** that will cause the error and describe the error in few lines. If a code segment produces garbage value represent it with "**G**". *Note: There is no syntax error in following code segments.*

Part (i)

```
-----------------------------------------------------
void function_B(int* &p, int *q)
{
	q = new int;
	*q = *p - 100;
	*p = *q - 100;
	delete q;
}
void function_A(int * p, int* &q)
{
	p = new int;
	*p = *q + 100;
	*q = *p + 100;
	function_B(p, q);
	delete p;

}

int main()
{
	int x = 1000;
	int* ptr1=&x;
	int* ptr2 =new int;
	*ptr2 = 600;
	cout << *ptr1 << " " << *ptr2 << endl;
	function_A(ptr1, ptr2);
	cout << *ptr1 << " " << *ptr2 << endl;
	function_B(ptr1, ptr2);
	cout << *ptr1 << " " << *ptr2;
	delete ptr2;
	return 0;
}
```

1000 600

1000 800

800 800

Part(ii)

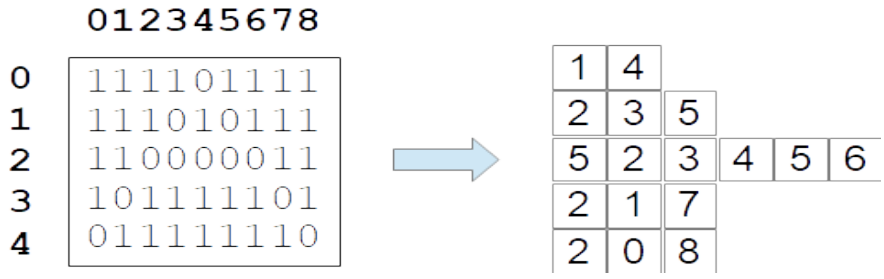
```
void Mystery(char *str1, char* str2){
int index = 0;
while (str2[index] != '\0')
{
       str1[index] = str2[index];
       index++;
       }
       str1[index] = '\0';
}
int main(){
 char str1[] = "Object Oriented Programming
Sessional-I";
 char str2[] = "Winter is Coming!";
 Mystery(str2, str1);
 cout << str2;
 return 0;
}
```

Length of str2 is less than str1, therefore while copying it will access illegal memory.

Object Oriented
PGGGGGGGGGGGGGGGGGGGGG

Question 2 **(10 points)**

Consider a black and white image that is represented as a matrix of 1's and 0's. We would like to compress the image by storing the index location of a specific value (either 0 or 1) for each row. For instance, the figure below shows transformation of an image in our desired encoding by storing location of 0's:



The first cell in each row of the result shows the number of locations with the specific value (0 or 1) in that row of the original matrix. Write a function that uses the following prototype for this transformation:

int compress(int** image, int rows, int columns, int value);**

Use dynamic memory allocation for the result. The parameter **value** specifies the value to use for compression (either 0 or 1).

```
int** compress(int** image, int rows, int columns, int value){
    int** compressedImage = new int*[rows];
    for(int i=0; i < rows; i++){
        int count = 0;
        for (int j=0; j < columns; j++){
             if (image[i][j] == value){
                count++;
            }
        }

        compressedImage[i] = new int[count+1];
        compressedImage[i][0] = count;

        for (int j=0,k=1; j < columns; j++){
            if (image[i][j] == value){
                compressedImage[i][k] = j;
                k++;
            }
        }
    }

    return compressedImage;
}
```

Question 3 **(15 points)**

Consider a polyline that comprises of multiple line segments connected together. Each line segment can be represented as a pair of x-y coordinates and two adjacent segments share a common coordinate i.e. a new line segment starts where the previous ends. A sample polyline is depicted below. Length of a polyline can be computed by summing up the length of each segment. Provide implementation for a C++ class such that it may be used to instantiate objects as given in the following driver program:

```
int main()
{
    int coordinates[][2] =
      {{1,0},{3,4},{2,4},{4,9}};
    Polyline line(coordinates,4);
    cout << line.length() << endl;
     // prints 10.85

   return 0;
}
```

Fig 3(a): A sample driver program

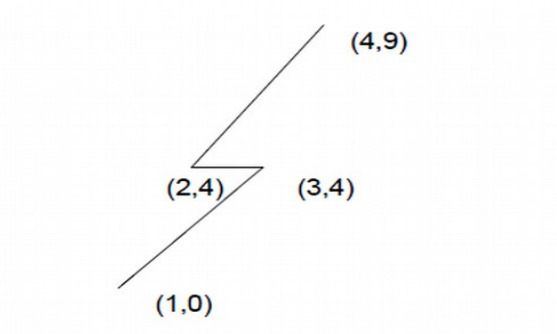


Fig 3(b): An illustration of an actual polyline

You must ensure the following in your answer:

- header and implementation are separate
- provide implementation of necessary constructor(s), destructor and member functions as given in the driver program
- use dynamic memory allocation to avoid unnecessary space.

Note: Length of a line segment can be calculated using Euclidean distance: $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$. You can use standard math library for your implementation.

// provide header definition here

```
class Polyline
{
    private:
        int (*coordinates)[2];
        int size;

    public:
        Polyline();
        Polyline(int coordinates[][2],int size);
        float length();
        ~Polyline();
};
```

// provide implementation here

```
Polyline::Polyline()
{
    coordinates = 0;
    size = 0;
}

Polyline::Polyline(int coordinates[][2], int size)
{
    this->size = size;
    this->coordinates = new int[size][2];
    for(int i=0; i < this->size; i++){
        this->coordinates[i][0] = coordinates[i][0];
        this->coordinates[i][1] = coordinates[i][1];
    }
}

float Polyline::length(){
    float result = 0;
    for(int i=0; i < this->size-1; i++){
        result += sqrt(pow(this->coordinates[i+1][0] - this->coordinates[i][0],2)+
                       pow(this->coordinates[i+1][1] - this->coordinates[i][1],2));

    }
    return result;
}

Polyline::~Polyline()
{
    if (this->coordinates != 0){
        delete [] this->coordinates;
    }
}
```