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:	**60 Minutes**	**Total Marks:**	**25**
Paper Date:	**26-Feb-2020**	**Page(s):**	**4**
Section:	**ALL**	**Section:**	
Exam:	**Midterm Exam 1**	**Roll No:**	

Instruction/Notes: Answer in the space provided
You can ask for rough sheets but **they will not be graded or marked**
In case of confusion or ambiguity make a reasonable assumption.
Questions are not allowed.

Question 1: (5+5 marks)

Write output against each of the following in proper format? (Write **G** for garbage value, if any).

Part(A) (5 marks)

```
void Interchange(int*& a, int*& b)
{
        int *temp = a;
        a = b;
        b = temp;
}
```

```
void print(int **a, int size)
{
        for (int **i = a; i < a + size; i++){
                for (int j = 0; j < size; j++)
                        cout << *(i[0] + j) << ",";
                cout<<endl;
        }
}
```

```
int main()
 {
        int size = 3;
        int ** a = new int *[size];
        for (int i = 0; i < size; i++)
        {
                *a = new int[size];
                for (int j = 0; j < size; j++){
                        *(*a + j) =i+ j + 1;
                }
                int index = (i + 1) % size;
                Interchange(*a, a[index]);
        }

        print(a, size);

        for (int i = 0; i < size; i++)
                delete[] a[i];
        delete[] a;
        a =nullptr;

return 0;
}
```

OUTPUT:

```
3,4,5,
1,2,3,
2,3,4,
Press any key to continue .
```

Roll Number:_____ Section: _____

Part(B) **(5 marks)**

```
class A {
        char c1, c2;
public:
        A()
        {     c1 = 'z';
              cout << c2 << ",";
        }

        A(char c)
        {             c2 = 'A';
                      cout << c << ",";
        }
        void setC1(char n1) { c1 = n1; }
        void setC2(char n2) { c2 = n2; }
        void swap() { c1 = c2; c2 = c1; }
        char getC1() { return c1; }
        char getC2() { return c2; }
};
```

```
void main() {
        A a1;
        A a2('F');

        cout << endl;
        cout << a1.getC1() << "," << a2.getC2() << endl;

        a1.setC1('O');
        a2.swap();

        cout << a1.getC1() << "," << a1.getC2() << endl;
        cout << a2.getC1() << "," << a2.getC2() << endl;

        a1.swap();
        cout << a1.getC1() << "," << a1.getC2() << endl;
}
```

OUTPUT:

```
╠,F,
z,A
O,╠
A,A
╠,╠
Press any key to continue . . .
```

Question 2: (3+3+9 marks)

You have to implement the C++ code of a function **splitArray,** that takes a dynamic square two-dimensional array ***Arr*** and its size **n** as parameters. Array ***Arr*** comprises of random integer numbers. The function **splitArray** will split array ***Arr*** in two sub arrays and return them along their sizes. The first sub array will contain all prime numbers of original ***Arr*** and second sub array will contain all non-prime numbers.

Sample run of **splitArray** is shown below:

Input Arr:

```
Enter a Number greater than 1: 5
Original Array:
38      19      38      37      55
97      65      85      50      12
53      0       42      81      37
21      45      85      97      80
76      91      55      6       57
```

Sub Arrays returned by splitArray:

```
Sub Array of Primes:
19      37
97
53      37
97

Sub Array of Non-Primes:
38      38      55
65      85      50      12
0       42      81
21      45      85      80
76      91      55      6       57
```

Note:

- The Original array **Arr** should remain intact (there should be no change in its size and data) after function call.
- The function **bool isPrime (int n)**, which returns true if a number is prime and false otherwise is already implemented. **So you do not need to implement it**.
- Your code should be free of dangling pointers and memory leak.

Part (A) Write down the function header of **splitArray**. **(3 marks)**

```
void splitArray(int ** Arr, int n, int **&s1, int &n1, int *&col1, int **&s2, int &n2, int *&col2);
```

Part (B) Write the C++ code of a generic function **deallocateArray**, which can deallocate memory of 2d arrays of any size. This function will be called from main function **three** times for deallocation of original array ***Arr***, and two sub arrays which are created and returned by function **splitArray**. **(3 marks)**

```
void deallocateArray(int ** s, int n){
	for (int i = 0; i < n; i++) //deallocate memory
		delete[] s[i];
	delete[]s;
	s = nullptr;
}
```

Part (C) Write the C++ code of function **splitArray**. **(9 marks)**

Roll Number:_____ Section: _____

```
void splitArray(int ** Arr, int n,
                int **&s1, int &n1, int *&col1,
                int **&s2, int &n2, int *&col2){
	n1 = 0; n2 = 0;
	// calculate row sizes of both arrays
	for (int i = 0; i < n; i++){
		int pcount = 0, npcount = 0;
		for (int j = 0; j < n; j++){
			if (is_prime(Arr[i][j]))
				pcount++;
			else
				npcount++;
		}
		if (pcount > 0) n1++;
		if (npcount > 0) n2++;
	}
	// for storage of column sizes of each row
	col1 = new int[n1];
	col2 = new int[n2];

	// calculate column sizes of both arrays
	int i1 = 0, i2 = 0;
	for (int i = 0; i < n; i++){
		int pcount = 0, npcount = 0;
		for (int j = 0; j < n; j++){
			if (is_prime(Arr[i][j]))
				pcount++;
			else
				npcount++;
		}
		if (pcount > 0)  col1[i1++] = pcount;
		if (npcount > 0) col2[i2++] = npcount;
	}

	//create new arrays
	s1 = new int*[n1];
	for (int i1 = 0; i1 < n1; i1++)
		s1[i1] = new int[col1[i1]];

	s2 = new int*[n2];
	for (int i2 = 0; i2 < n2; i2++)
		s2[i2] = new int[col2[i2]];

	//copy data in splited arrays
	for (int i = 0, i1 = 0, i2 = 0; i < n; i++, i1++, i2++){
		for (int j = 0, c1 = 0, c2 = 0; j < n; j++){
			if (is_prime(Arr[i][j])){
				if (i1 < n1 && c1 < col1[i1]) s1[i1][c1++] = Arr[i][j];
			}
			else{
				if (i2 < n2 && c2 < col2[i2]) s2[i2][c2++] = Arr[i][j];
			}
		}
	}
}
```