Serial No:

1st Sessional Exam
Total Time: 1 Hour
Total Marks: 50

____
Signature of Invigilator

CS1004 Object Oriented Programming
Tuesday, February 28, 2023
Course Instructor
Dr. Bilal Khan, Dr Imran Babar, Dr. Khalid Hussain Mr. Rizwan ul Haq, Mr. Saud Arshad, Mr. Ch. Usman Ghous

Roll No	Section	Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Verify at the start of the exam that you have a total of five (5) questions printed on ten (10) pages including this title page.
2. Attempt all questions on the question-book and in the given order.
3. The exam is closed books, closed notes. Please see that the area in your threshold is free of any material classified as 'useful in the paper' or else there may a charge of cheating.
4. Read the questions carefully for clarity of context and understanding of meaning and make assumptions wherever required, for neither the invigilator will address your queries, nor the teacher/examiner will come to the examination hall for any assistance.
5. Fit in all your answers in the provided space. You may use extra space on the last page if required. If you do so, clearly mark question/part number on that page to avoid confusion.
6. Use only your own stationery and calculator. If you do not have your own calculator, use manual calculations.
7. Use only permanent ink-pens. Only the questions attempted with permanent ink-pens will be considered. Any part of paper done in lead pencil cannot be claimed for checking/rechecking.

	Q-1	Q-2	Q-3	Q-4	Q-5	Total

National University of Computer and Emerging Sciences

Department of Computer Science **Chiniot-Faisalabad Campus**

Total Marks	5	15	5	20	5	**50**
Marks Obtained						

Vetted By: ____________________________**Vetter Signature:** _____________________

University Answer Sheet Required: **No** ☐ **Yes** ☐

Question 1:	Marks: 2.5 + 2.5 = 5

a. What is memory leak? Explain with a small piece of code.

When a pointer is allocated dynamic memory but without deallocating or keeping the pointer of the memory this pointer is allocated another memory or pointed to some other location. The memory previously allocated is no more accessible. It is called memory leak. **(1)**

```
int main()              (1.5)
{
      int num;
      int *iPtr = new int[20];
      iPtr = new int[20]; //Memory Leak
      iPtr = &num; //Memory Leak
}
```

b. What is meant by a dangling pointer? Give an example of code to explain.

When a pointer is pointing to an invalid memory location it is called dangling pointer. i.e. a pointer allocated a dynamic memory and then deallocated the memory but didn't made the pointer null. (1)

```
int main()                    (1.5)
{
      int *iPtr = new int[20];
      delete[] iPtr; //iPtr is a dangling pointer at the moment.
      iPtr = nullptr; //Solution to dynamic pointer
}
```

Question 2:	Marks: 4 + 3 + 5 + 3 = 15

Assume a double pointer **(dPtr)** of type double, and two variables, **rows** and **cols** to hold the dimensions of the dynamic array are declared in **main()**.

Now perform the following tasks.

Note: Every function should clearly show all the function parameters. Make sure that the rows and cols are updated properly when required with in the called function not main().

1. Create a function **(allocateMem)** that will allocate a memory of **rows * cols** to the pointer. Rows and cols must be input by the user in this function. **(4 marks)**

```
int** allocateMem(int& row, int& col) //row and col must be pass by ref (1)
{
	//Input of row and col          (1)
	cout << "Enter the Rows of the matrix : ";
	cin >> row;
	cout << "Enter the Columns of the matrix : ";
	cin >> col;
	ptr = new int*[row];                    (1)
	for (int i = 0; i < row; ++i)           (1)
		ptr[i] = new int[col];
	return ptr;
}
```

OR

```
//double pointer row and col must be pass by ref (1)
void allocateMem(int**& ptr;  int& row, int& col)
{
	//Input of row and col          (1)
	cout << "Enter the Rows of the matrix : ";
	cin >> row;
	cout << "Enter the Columns of the matrix : ";
	cin >> col;
	ptr = new int*[row];                    (1)
	for (int i = 0; i < row; ++i)           (1)
		ptr[i] = new int[col];
}
```

2. Create a function **(getInput)** that will be called from main() and it will take input from the user in this dynamically allocated array. **(3 marks)**

```
//const can be omitted
void getInput(int **ptr, const int row, const int col)      (1.5)
{
	for (int i = 0; i < row; ++i)           (1.5)
		for (int j = 0; j < col; ++j)
			cin >> ptr[i][j];
}
```

3. Now create a function **(extendSize)**, that will make the size of the dynamically allocate array double the size it already had. Make sure it should preserve the data already in the array. **(5 marks)**

```
void extendSize(int& **ptr;  int& row, int& col)                    (1)
{
        int tRows = row * 2;            //Temporary declarations    (1)
        int tCols = col * 2;
        int** temp;
        temp = new int*[tRows];
        for (int i = 0; i < row; ++i)
                temp[i] = new int[tCol];
        for (int i = 0; i < tRows; ++i)  //Allocating memory of larger size and
                                        //copying data              (1.5)
        {
                for (int j = 0; j < tCols; ++j)
                {
                        if (i < row && j < col)
                                temp[i][j] = ptr[i][j];
                        else
                                temp[i][j] = 0;
                }
        }
        for (int i = 0i < row; ++i)      //Deallocating old memory
                delete[] ptr[i];                                    (0.5)
        delete ptr;
        ptr = temp;             //Setting pointer to point to new memory   (1)
        row = tRow;             //Setting row to new size
        col = tCol;             //setting col to new size
}

                                        OR
int** extendSize(int** ptr;  int& row, int& col)
{
        int tRows = row * 2;
        int tCols = col * 2;
        int** temp;
        temp = new int*[tRows];
        for (int i = 0; i < row; ++i)
                temp[i] = new int[tCol];
        for (int i = 0; i < tRows; ++i)
        {
                for (int j = 0; j < tCols; ++j)
                {
                        if (i < row && j < col)
                                temp[i][j] = ptr[i][j];
                        else
                                temp[i][j] = 0;
                }
        }
        for (int i = 0i < row; ++i) //Deallocating old memory
                delete[] ptr[i];
        delete ptr;
        row = tRow;             //Setting row to new size
        col = tCol;             //setting col to new size
        return temp;  //Finally return the new memory's address
}
```

4. Finally make a function (**deallocateMem**). This function will be passed this dPtr and it will deallocate the memory, allocated to this pointer. **(3 marks)**

```
void deallocateMem(int** ptr/*int** &ptr*/;  int row)      (1)
{
      for (int i = 0; i < row; ++i) //Deallocating old memory   (1)
            delete[] ptr[i];
      delete ptr;            (1)
}
```

Question 3:	Marks: 5

A phone number, such as (212) 767-8900, can be thought of as having three parts: the area code (212), the exchange (767), and the number (8900). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure phone. Create two structure variables of type phone. Initialize one, and have the user input a number for the other one. Then display both numbers. The interchange might
look like this:

Enter your area code, exchange, and number: 415 555 1212
My number is (212) 767-8900
Your number is (415) 555-1212

```
struct phone {
	string areaCode = "\0";
	string exchange = "\0";
	string number = "\0";
};

int main() {
	phone myPhone = { "212", "767", "8900" };
	phone yourPhone;

	cout << "Enter your area code : ";
	getline(cin, yourPhone.areaCode);
	cout << "Enter your exchange : ";
	getline(cin, yourPhone.exchange);
	cout << "Enter your number : ";
	getline(cin, yourPhone.number);
	cout << endl;

	cout << "My number is (" << myPhone.areaCode << ") "
	<< myPhone.exchange << "-" << myPhone.number << endl;
	cout << "Your number is (" << yourPhone.areaCode << ") "
	<< yourPhone.exchange << "-" << yourPhone.number << endl;



}
```

Question 4:	Marks: 5 + 5 + 5 + 5 = 20

Write a program that reads students' names followed by their test scores. The program should output each student's name followed by the test scores and the relevant grade. It should also find and print the highest test score and the name of the students having the highest test score.

Student data should be stored in a struct variable of type studentType, which has four components: studentFName and studentLName of type string, testScore of type int (testScore is between 0 and 100), and grade of type char. Suppose that the class has 20 students. Use an array of 20 components of type studentType.

Your program must contain at least the following functions:

a. A function to read the students' data into the array.
b. A function to assign the relevant grade to each student. **(For score less than 50 grade is F otherwise P)**
c. A function to find the highest test score.
d. A function to print the names of the students having the highest test score.

Your program must output each student's name in this form: last name followed by a comma, followed by a space, followed by the first name.

```
const int MAX_STUDENTS = 20;

struct studentType {
string studentFName;
string studentLName;
int testScore;
char grade;
};
//Reading data from file
void readData(studentType students[], int numStudents) {
ifstream obj;
obj.open("File.txt");
if (!obj.is_open()) {
cout << "File is not open\n";
}
int i = 0;
char ch;
while (!obj.eof()) {
if (!obj.eof()) {

obj >> students[i].studentFName;
obj >> students[i].studentLName;
obj >> students[i].testScore;
i++;
}
}

}


//Assigning the grades
void assignGrades(studentType students[], int numStudents) {
for (int i = 0; i < numStudents; i++) {
if (students[i].testScore < 50) {
students[i].grade = 'F';
}
else {
students[i].grade = 'P';
}
}
}

//Finding highest test score
int findHighestScore(studentType students[], int numStudents) {
int highestScore = -1;
for (int i = 0; i < numStudents; i++) {
if (students[i].testScore > highestScore) {
```

```
highestScore = students[i].testScore;
}
}
return highestScore;
}

//Printing names of students with highest score
void printNamesOfHighestScorers(studentType students[], int numStudents,
int highestScore) {
cout << "Highest test score: " << highestScore << endl;
cout << "Students with highest test score:\n";
for (int i = 0; i < numStudents; i++) {
if (students[i].testScore == highestScore) {
cout << students[i].studentLName << ", " << students[i].studentFName <<
endl;
}
}
}
```

Question 5:	Marks: 5

Write a C++ program that uses pointer arithmetic to determine the size of a variable for any data type, to which it is pointing to.

```
int i = 10;
char c = 'a';
float f = 2.002;
double d = 2.35127;

//allocatng pointer to the variable
int *iptr = &i;
char *cptr = &c;
float *fptr = &f;
double *dptr = &d;

//using pointer to determine the size of variable
cout << "Size of int variable : " << sizeof(*iptr) << endl;
cout << "Size of char variable : " << sizeof(*cptr) << endl;
cout << "Size of float variable : " << sizeof(*fptr) << endl;
cout << "Size of double variable : " << sizeof(*dptr) << endl;
```