

Computer Programming

Final Exam, Spring 2015

~~Date: 8th June 2015~~ ~~Marks: 70~~ ~~Time: 3 hrs.~~

Question 1 (10 + 10 = 20 points)

Constraint: For this question your code cannot create an extra array (not even a onedimensional array) to accomplish the given task. But you can create one or two extra integers.

No credit if this constraint is violated.

- a) Write a function called `rotateOuterLayerBy1`, which accepts a two dimensional square matrix and its dimension, `n`, and rotates the outermost layer by one place, anti-clockwise. Note the following example:

A 4x4 array Outer layer rotated once

0	1	2	3
4	5	6	7
8	9	1	1
1	1	1	1
2	3	4	5

1	2	3	7
0	5	6	1
4	9	1	1
8	1	1	1
2	3	4	5

Solution

```
void rotateOuterLayerBy1(int **matrix, int rows, int cols){
    int i=0,j=0; int temp=0;

    temp=matrix[0][0];

    //0th row shifted left
    for(i=0;i<cols-1;i++)
        matrix[0][i]=matrix[0][i+1];

    //last column shifted up
    for(i=0;i<rows-1;i++)
        matrix[i][cols-1]=matrix[i+1][cols-1];

    //last row shifted right
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
for(i=cols-1;i>0;i--)  
    matrix[rows-1][i]=matrix[rows-1][i-1];
```

```
//first column shifted down
```

```
for(i=rows-1;i>0;i-)  
    matrix[i][0]=matrix[i-1][0];  
  
matrix[1][0]=temp;  
}
```

- b) Write a function called `rotateImage90`, which accepts an image matrix, i.e. a two dimensional square matrix, and its dimension, `n`, and rotates the entire image by 90 degrees anti-clockwise. Following is an example of a 4x4 image and its 90° rotated form.

A 4x4 image

0	1	2	3
4	5	6	7
8	9	1	1
1	1	1	1
2	3	4	5

90 degree rotation

3	7	1	1
2	6	1	1
1	5	9	1
0	4	8	1
			2

Hint: You can smartly use `rotateOuterLayerBy1` and 2D pointers to accomplish this task. **Solution**

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
void rotateOuterLayerBy1(int **matrix, int rows, int cols, int rNo, int cNo){
    int i=0,
    j=0; int
    temp=0;
    if(rows!=0 && cols!
    =0){
        temp=matrix[rNo][cNo];

        //0th row shifted left    for(i=cNo;i<cols-
cNo-1;i++)
            matrix[rNo][i]=matrix[rNo][i+1];

        //last column shifted up
        for(i=rNo;i<rows-rNo-1;i++)            matrix[i]
[cols-cNo-1]=matrix[i+1][cols-cNo-1];

        //last row shifted right    for(i=cols-1-
cNo;i>cNo;i--)            matrix[rows-rNo-1]
[i]=matrix[rows-rNo-1][i-1];
```

```
        //0th column shifted down    for(i=rows-rNo-
1;i>cNo;i--)
            matrix[i][cNo]=matrix[i-1][cNo];

        matrix[rNo+1][cNo]=temp;
    }
}
```

```
int main(){

    int r=4;
    int c=4;
    int **arr=new int*[r];
    int i=0, j=0;
    int n=0;

    for(i=0;i<r;i++)
        arr[i]=new int[c];

    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            arr[i][j]=n++;

    for(i=0;i<3;i++){
```

Computer Programming

Final Exam, Spring 2015

~~Date: 8th June 2015~~

~~Marks: 70~~

~~Time: 3 hrs.~~

```
        rotateOuterLayerBy1(arr, r, c,0,0);
        //rotateOuterLayerBy1(arr, r, c,1);
    }
    rotateOuterLayerBy1(arr, r, c,1,1);
    for(int i=0; i<r;i++){
        for(int j=0;j<c;j++){
            cout<<arr[i][j]<<" ";
            cout<<endl;
        }

        return 0;
    }
}
```

Question 2 (7.5 + 7.5 = 15 points)

Consider a class called **Student** that has the following private data members: name (char*), that has an arbitrary length, a cgpa e.g. 3.5 and an object of class **Address**. Address itself is a class having two data members: a p.o. box number e.g. 6550 and a home address (char*) e.g. 852-B Faisal Town Lahore.

A class **ClassRoom** contains a dynamically allocated array of pointers to dynamically allocated Student objects.

- Write the destructor for the class **ClassRoom**, and necessary methods in the other classes so that there are no memory leaks.
- Considering the following main program, write down all necessary functions in all three of the above classes.

```
int main()
{
    ClassRoom a;
    //student data is read from a file
    ClassRoom c=a.warningStudents();
    //c contains new copies of students with
    //cgpa<2 from classRoom a
    return 0;
}
```

~~Note: Pay attention to adding exactly the methods required to run the above code without memory management problems. You don't have to write the method warningStudents.~~

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
class address{
private:
    char *homeAddress;
    int poxNumber;
public:
    address(){
        homeAddress=NULLPTR;
        poxNumber=0;
    }
    address(const address &a){
        if(a.homeAddress!=NULLPTR){
            homeAddress=new char[strlen(a.homeAddress)+1];
            strcpy(homeAddress, a.homeAddress);
        }
        else
            homeAddress=NULLPTR;
        poxNumber=a.poxNumber;
        cout<<"add copy const\n";
    }
    ~address(){
        if(homeAddress!=NULLPTR)
            delete []homeAddress;
    }
};

class student{
private:
    char *name;
    float cgpa;
    address ad;
public:
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
        student():ad(){
            name=nullptr;
            cgpa=0;
        }
        student(const student &s):ad(s.ad){
            if(s.name!=nullptr)
            {
                name=new char[strlen(s.name)+1];
                strcpy(name, s.name); }
            else
            {
                name=nullptr;
                cgpa=s.cgpa;
                cout<<"std copy const\n";
            }
            ~student(){
                if(name!=nullptr)
                    delete []name;
            }
        };

class classRoom{
private:
    student **s;
    int size;
public:
    classRoom(){
        size=1;
        s=new student*[size];
        for(int i=0;i<size;i++)
            s[i]=new student();
    }
    ~classRoom(){
        for(int i=0;i<size;i++)
            delete s[i];

        delete []s;
        s=nullptr;
        size=0;
    }

    classRoom warningStudents(){
        classRoom *r=new classRoom();
        return *r;
    }

    classRoom(classRoom &c){
        size=c.size;
    }
};
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
s=new student*[c.size];
for(int i=0;i<size;i++)
s[i]=new student(*(c.s[i]));

    //c.s[i]=s[i]; acceptable
}

};
int main(){
    classRoom a;
    //student data is read from a file
    classRoom c=a.warningStudents();
    //c contains new copies of students
with
    //cgpa<2 from classRoom a
    return 0;
}
```

Question 3 (10 + 5 = 15 points)

- a) Write a function template called `removeAll` which accepts a dynamic array of any type of objects, its size, and an object called `key`. It returns a new array with all remaining elements after all instances of `key` have been removed from the original array. The function should also specify the size of this newly created array. Note: `specify` does not mean `print`. **Solution**

```
template<typename T>
T* removeAll(T *arr, int &size, T key){

    int newSize=0;
    for(int i=0;
i<size;i++)
        if(arr[i]!=key)
            newSize++;

    T *narr=new T[newSize];

    for(int i=0, j=0; i<size;i++){
        if(arr[i]!=key){
            narr[j]=arr[i];
            j++;
        }
    }
    size=newSize;
    return narr;
}
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

}

- b) We want removeAll to also work when key is a pointer to an object. In this case it should compare not the pointer but the data with the elements of the array. How will you solve this problem? Indicate the change in code.

Solution

```
template<typename T>
T* removeAll(T *arr, int &size, T *key){

    int newSize=0;
    for(int i=0; i<size;i++){
        if(arr[i]!=*key)
            newSize++;

    T *narr=new T[newSize];

    for(int i=0, j=0; i<size;i++){
        if(arr[i]!=*key){
            narr[j]=arr[i];
            j++;
        }
    }
    size=newSize;
    return narr;
}
```

Question 4 (20 points)

Design and develop a program with three levels of classes to maintain personal files for HR department in a university. There are two categories of people in the university, staff and students. A special category of staff members are executives. And students may be graduates or undergraduates. At the top level everybody, including students and staff, is a person, and every person has information such as: name (char*), gender (bool), and age (int). All staff members have department (such as management, finance, etc.), and scale (from 0 to 5). An executive staff member includes data regarding the department managed by the executive, which includes: the department budget in rupees, and number of employees in the department. All students maintain a major (char*), such as CS, EE etc., and a gpa (float). Every graduate-student has a research area (such as AI, Software Engineering etc.), and number of specialization courses taken, whereas, every undergraduate has credits earned, which is the number of credit hours earned by them. Furthermore, we can compute the merit of any student. The merit of an ungraduated is the product of his gpa and credits earned, whereas the merit of a graduate student is zero if they have gpa less

Computer Programming

Final Exam, Spring 2015

~~Date: 8th June 2015~~ ~~Marks: 70~~ ~~Time: 3 hrs.~~
than 2.5, else, it is the product of their gpa and the number of specialization courses taken by them.

Note the following:

- Add appropriate constructors with parameters to all classes in the hierarchy. Use member initialization lists where possible.
- Detect the methods to be added to these classes by looking at the following main program and the corresponding output. Add polymorphism where required. Note: pay special attention to which functions should be virtual and which pure virtual, etc.
- All strings are allocated char* type class members. You cannot use the sting class. Make sure that there are no memory leaks anywhere in the program.

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
void main(){
    Person ** everybody;
    int n;
    //data is read into array from a
    file //n contains the total number of
    persons    cout<<"Total number of Persons:
    "<<n<<endl;    for(int i=0;i<n;i++)
        //print information on screen    everybody[i]-
    >printInfo();

    cout<<endl<<endl;

    Students ** allStudents; int
    m;
    //data is read into array from a
    file //m contains the total number of
    students    cout<<"Total number of
    Students:"<<m<<endl;    for(int i=0;i<n;i++)
        cout<<"Merit of this Student is:"
        <<allStudents[i]->computeMerit();

    //clean up all memory
    for(int i=0; i<n; i++)
    delete everybody[i];    for(int
    i=0; i<m; i++)    delete
    allStudents[i];

    delete [] everybody;
    delete [] allStudents;
}
```

//Output

Total number of Persons: 5

Ali Hayat, Male, 19, CS, 3.45, Undergrad with 35 credits
--

Wajeeja Ali, Female, 31, Accounts, Scale 2
--

Fatima Ahmed, Female, 20, EE, 3.21, Graduate working in AI, Specialization Courses: 4 Wazir Khan, Male, 40, Management, Scale 5, Executive with 50 employees
--

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

Total number of Students: 2

Merit of this Student is: 13.45

Merit of this Student is: 15.25

```
class person{
protected:
    char *name;
    bool gender;
    int age;
public:
    person(){
        name=nullptr;
        gender=0;
        age=0;
    }
    person(char *n, bool g, int a){
        if(strlen(n)!=0){
            name=new char[strlen(n)+1];
            strcpy(name,n);
        }
        else
            name=nullptr;
        gender=g;
        age=a;
    }
    virtual void printInfo(){
        cout<<"name: "<<name<<endl;
        cout<<"gender: "<<gender<<endl;
        cout<<"age: "<<age<<endl;
    }
    virtual ~person(){
        delete []name;
    }
}; class staff:public
person{ protected:
    char *depart;
    int scale;
public:
    staff(){
        depart=nullptr;
        scale=-1;
    }
}
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
staff(char *n, bool g, int a, char *d, int
s):person(n,g,a){
    if(strlen(d)!=0){
        depart=new char[strlen(d)+1];
        strcpy(depart,d);
    }
    else
        depart=nullptr;
    scale=s;

}
void printInfo(){
    person::printInfo();
    cout<<"department: "<<depart<<endl;
    cout<<"scale: "<<scale<<endl;
}
~staff(){
    delete []depart;
}
}; class executives:public
staff{ protected:
    float budget;
    int noOfEmp;
public:
    executives(){
        budget=-1;
        noOfEmp=-1;
    }
    executives(char *n, bool g, int a, char *d, int s, float b, int e):staff(n, g,
a, d, s){
        budget=b;
        noOfEmp=e;
    }
    void printInfo(){
        staff::printInfo();
        cout<<"budget: "<<budget<<endl;
        cout<<"number of employees: "<<noOfEmp<<endl;
    }
    ~executives(){}
}; class student:public
person{ protected:
    char *major;
    float gpa;
    float merit;
public:
    student(){
        major=nullptr;
        gpa=0;
        merit=0;
    }
}
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

<pre> } student(char *n, bool g, int a, char *m, float cg, float mt)</pre>

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
        :person(n,g,a)
    {
        if(strlen(m)!=0){
            major=new char[strlen(m)+1];
            strcpy(major,m);
        }
        else
            major=nullptr;
        gpa=cg;
        merit=mt;
    }
    void printInfo(){
        person::printInfo();
        cout<<"major: "<<major<<endl;
        cout<<"gpa: "<<gpa<<endl;
        cout<<"merit: "<<merit<<endl;
    }
    virtual float computeMerit()=0;
    ~student(){
        if(major!=nullptr)
            delete []major;
    }
};
class gradStudent:public student{
protected:
    char *resrchArea;
    int noOfcourses;
public:
    gradStudent(){
        resrchArea=nullptr;
        noOfcourses=0;
    }
    gradStudent(char *n, bool g, int a,
char *m, float cg, float mt, char *ra, int cour):
        student(n,g,a,m,cg,mt)
    {
        if(strlen(ra)!=0){
            resrchArea=new char[strlen(resrchArea)+1];
            strcpy(resrchArea,ra);
        }
        else
            resrchArea=nullptr;
        noOfcourses=cour;
    }
    void printInfo(){
        student::printInfo();
```

Computer Programming

Final Exam, Spring 2015

Date: 8th June 2015

Marks: 70

Time: 3 hrs.

```
        cout<<"research area: "<<resrchArea<<endl;
        cout<<"no of courses: "<<noOfcourses<<endl;

    }
    float computeMerit(){
        if(gpa<2.5)
            return 0;
        else
            return gpa*noOfcourses;
    }
    ~gradStudent(){
        delete []resrchArea;
    }
}; class undergradStudent:public
student{ protected:
    int credits;
public:
    undergradStudent(){
        credits=-1;
    }
    undergradStudent(char *n, bool g, int a, char *m, float cg, float mt, int cr):
        student(n,g,a,m,cg,mt)
    {
        credits=cr;
    }
    void printInfo(){
        student::printInfo();
        cout<<"credits: "<<credits<<endl;
    }
    float computeMerit(){
        return gpa*credits;
    }
    ~undergradStudent(){}
};
```

Good Luck!