Computer Programming
Mid 2 – Spring 2014

Maximum Marks: 30 **16th April, 2014** **Time Allowed: 90 minutes**

Question 1: (2x5)

Give short and succinct answers to the following questions.

a. Write three distinct situations in which copy constructor of a class is called. Give an example for each.
Copy constructor is called in three situations
Passing an object as parameter by value e.g. AddFraction(fraction f1, fraction f2)
Return an object e.g.
fraction res;
return res ;
Initialization at the time of declaration e.g
Fraction f1, fraction f2(f1);

b. Is the piece of code below correct? If yes, what is the output of the function `Bar`. If no, why not?

```
class Foo {
int x;
static int count;
public:
      static int Bar(int i) {
              return x*i*count + x*i + 1;
      }
};
```

No it is not correct a static function cannot use non static data member as x is used here

c. How do we initialize static members of a class? Give an example.
int Foo::count=0;

d. What are the uses of "member initialization list"? Illustrate by giving examples.
Member initialize list is used to initialize const data members and to call the overloaded constructor of contained class object

e. What is a friend function? How do we declare it?
A friend function can access the private data members of a class and we declare them in the class using the key word friend

Question 2: (10)

Given a cstring (a null terminated character array); two indices, i and j, where i ≤ j, and a number n; write a C++ function that rotates the contents of the part of the cstring between i and j, n times towards right. For example, if the cstring is, "now this will be rotated", and i=4, j=7, and n=3, the cstring will become: "now hist will be rotated'.

Solution:

```
void rotate(char *str,int i, int j, int n){
        for(int k=0;k<n;k++){
                char temp = str[j];
                for(int a=j;a>i;a--)
                        str[j] = str[j-1];
                str[i] = temp;
        }
}
```

Question 3: **(10)**

Define a class Vector which can store a list of n>=1 integers. The behavior (functions) of the class must be defined according to the following driver function.

```
int main()
{
        Vector v1, v2(5), v3(5), v4(6);
        cin>>v2;
        cin>>v3;
        cin>>v4;
        v1 = v2 + v3 + v4;
        v2 = v3 = 5*v1;
        cout<<v1*5<<v2;
}
```

The statement **vector v2(5);** must construct a vector having 5 components and all its components must be initialized to zero. ***cin>>v2***; gets input for each component of ***v2*** from the user. The number of input components must be exactly equal to the number of components provided during construction (in this example 5 components). The addition of two vectors gives a resultant vector in which corresponding components are added. A vector can be multiplied either with an integer or with another vector. Let ***v1, v2*** be the two vectors and ***a*** an integer. The result of ***a*v1*** is a vector obtained by multiplying each component of ***v1*** by ***a***, ***v1*v2*** is the dot product of ***v1*** and ***v2***. Note that in both addition and multiplication, the value of missing components must be considered 0. Consider the following example

Let $v1=\begin{bmatrix}1\\2\\3\end{bmatrix} \wedge v2=\begin{bmatrix}4\\5\end{bmatrix}$, the result of ***5*v1*** or ***v1*5*** is $\begin{bmatrix}5\\10\\15\end{bmatrix}$, $v1+v2=\begin{bmatrix}5\\7\\3\end{bmatrix}$ and

v1*v2 = (1*4)+(2*5)+(3*0)=14

```
class vector{

        friend vector operator*(int , vector);
        friend ostream& operator<<(ostream &out, vector v);
        friend istream& operator>>(istream &out, vector &v);
        int n;
        int *comp;
private:
        vector(){
                n=0;
                comp=null;
        }
        vector(int a){
                n=a;
                comp = new int[n];
                for(int i=0;i<n;i++)
                        comp[i] = 0;
        }
        vector(vectoe &copy){
                n= copy.n;
                comp = new int[n];
                for(int i=0;i<n;i++)
                        comp[i] = copy.comp[i];
        }
        vector & operator=(vector &rhs){
                if(&rhs != this){
                        if(comp!=null) delete [] comp;
                        n=rhs.n;
                        comp = new int [n];
                        for(int i-0;i<n;i++)
                                comp[i] = rhs.comp[i];
                }
        }
        ~vector(){
                if(comp!=null) delete [] comp;
        }

        int operator*(vector v){
                int min,sum = 0;
                if(n<v.n)
                        min = n;
                else
                        min = v.n;
                for(int i=0;i<min;i++)
                        sum += comp[i]*v.comp[i];
                return sum;
        }

        vector operator*(int a){
                vector res(n);
                for(int i=0;i<n;i++)
                        res.comp[i] = a*comp[i];

                return res;

        }

        vector operator+(vector v){

                int min,max;
                if(n<v.n){
                        min = n;
                        max = v.n;
                }
                else{
                        min = v.n;
                        max = n;
                }

                vector res(max);

                for(int i=0;i<min;i++)
                        res.comp[i] = comp[i] + v.comp[i];
                return res;
        }
};

vector operator*(int a, vector v){
                vector res(v.n);
                for(int i=0;i<n;i++)
                        res.comp[i] = a*v.comp[i];

                return res;

        }

ostream& operator<<(ostream &out, vector v){
        for(int i=0;i<v.n;i++)
                out<<v.comp[i]<<"\t";
        out<<endl;
        return out;

}

istream& operator>>(istream &in, vector &v){
        for(int i=0;i<v.n;i++)
                in>>v.comp[i];
}
```