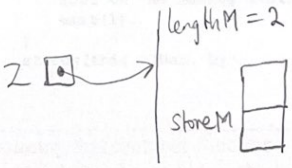
Question 1(9 marks) – Object Oriented Basic Elements

Consider the following C++ program. If you run this program, when some of the member functions, constructors, assignment operator, destructor, etc., are called, several lines of output will be created. In the following, write clearly what is the output of the program. **Please pay attention to the proper order of outputs. Marks will not be awarded for incorrect order of the output.**

```
#include <iostream>

using namespace std;

class Vector {
public:
  Vector(int n = 10);
  ~Vector();
  Vector(const Vector& source);
  Vector& operator =(const Vector& rhs);
private:
    int lengthM; // number of elements in storeM excluding
    int* storeM; // a pointer to the beginning of an array
                 // of integer, allocated dynamically.
};

Vector::Vector(int n): storeM(new int[n]), lengthM(n){
    cout << "\n ctor is called. ";
}

Vector::Vector(const Vector& source):
lengthM(source.lengthM), storeM(new int[source.lengthM+1]){
    for(int i= 0; i < lengthM; i++) storeM[i] = source.storeM[i];
    cout << "\n copy ctor is called. ";
}

Vector::~Vector(){
    delete [] storeM;
    cout << "\n dtor is called. ";
}

Vector& Vector::operator =(const Vector& rhs){
    if(this == &rhs)
        return *this;
    delete [] storeM;
    lengthM = rhs.lengthM;
    storeM = new int [lengthM];
    for(int i= 0; i < lengthM; i++) storeM[i] = rhs.storeM[i];
    cout << "\n operator =. ";
    return *this;
}

void fun( Vector& a, const Vector b){
    Vector *z = new Vector(2);
    a = b;
    delete  z;
}

int main(void){
    Vector x[2];
    fun(x[0], x[1]);
    return 0;
}
```

x[0] n=10 , x[1] n=10



ctor is called. ✓
dtor is called. X
Operator=. X
dtor is called. X
dtor is called. ✗
dtor is called. ✓

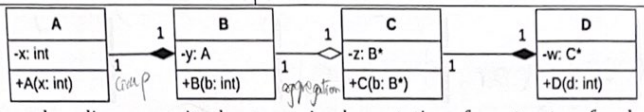
ctor
ctor
copy ctor
ctor
dtor called
operator=.
dtor
dtor
dtor
dtor

Question 2(6 marks) – Aggregation and Composition

Consider partial definition of classes A, B, C, D, and the following class diagram:

```
class A {
    int x;
public:
    A(int x): x(x){}
    int get_x()const {return x;}
};

class B {
    A y;
public:
    B(int b);
    A get_A()const {return y;}
};
```

```
class C {
    B* z;
public:
    C(B* b);
    const B* get_B()const {return z;}
};

class D {
    C* w;
public:
    D(int d);
};
```



Based on the given class diagram, write the proper implementation of constructors for classes: B, C, and D, in the following space:

Class B

```
B::B(int b){
    y = new A(b);
}
```

0.5

Class C

```
C::C(B* b){
    z = b;
}
```

2

Class D

```
D::D(int d){
    C* c;
    w = c;
    w = new C(new B(d));
}
```

0.1

Question 3 (10 marks) – Inheritance

Consider the partial definition of class Quux, and class Bar that is derived from Quux.

```
// quux.h
class Quux {
protected:
    char *quux;
public:
    Quux(const char* q);
    ~Quux() {delete [] quux;}
    Quux(const Quux& src);
    Quux& operator = (const Quux& rhs);
};
```

```
// bar.h
class Bar: public Quux {
protected:
    char *bar;
public:
    Bar(const char* b, const char *q);
    ~Bar() {delete [] bar;}
};
```

```
// quux.cpp

Quux::Quux(const char* q): quux(new
char[strlen(q)+1]) {
    if(quux == nullptr) {
        cout << "No memory available...";
        exit(1);
    }
    strcpy(this ->quux, q);
}
```

```
// bar.cpp
Bar::Bar(const char* b, const char* q):
Quux(q),bar(new char[strlen(b)+1]) {
    if(bar == nullptr) {
        cout << "No memory available...";
        exit(1);
    }
    strcpy(this ->bar, b);
}
```

Based on partial code given for both classes, and assuming that member functions, copy constructor, and assignment operator for class Quux is fully implemented, in the following space write ONLY the implementation of copy constructor and assignment operator for class Bar.

```
Bar::Bar(const Bar& src): Quux(src.bar){
    bar = new char[strlen(src.bar)+1];
    assert(bar != nullptr);
    strcpy(bar, src.bar);
}
```

0.4

Continues on back page..

Question 4 (4 marks) – Non-public Derivation of Classes

3-34

Consider the partial definition of the following classes. You may assume classes A, B, and C are defined in the same file in the given order from left to right.

```
class A{
public:
    int a1;

protected:
    int a2;
private:
    int a3;
};
```

```
class C: private A{
public:
    int c1;
    void funC();
protected:
    int c2;
private:
    int c3;
};
```

```
class D: public C
{
public:
    int d1;
    void funD();
protected:
    int d2;
private:
    int d3;
};
```

In the following table if a data member is **NOT** accessible in member functions `funC` and `funD`, mark it with (**X**), otherwise leave it blank. Negative marks will be given to wrong answer. Negative marks will be considered for wrong answers.

	a1	a2	a3	c1	c2	c3	d1	d2	d3
funC			X				X	X	X
funD	X	X	X			X			

4 - 3x0.22

Question 5 (6 marks)- Operator Overloading:

Consider the partial definition of class Text and the following C++ program. Then, answer the following question. You may assume all necessary header files are included and all member functions are properly implemented.

```
class Text{
    char* storageM;
    int lengthM;
public:
    Text(int n);
    /* REQUIES: n > 1;
     * PROMISES: allocates an array
     * of n characters on the heap and
     * sets '\0' to all of them.
     */
    Text(char* s);
    // Add function prototypes here
    friend istream& operator >>
        (istream& is, const Text& t);
    char& operator[](int index);
};
```

1 1

```
int main(){
  Text q = 20;
  q[1] = 'M';
  cout << "please enter word less than 20
          character: ";
  cin >> q;
  cout << q[0] << endl;
  return 0;
}
```

2

Question: In the following space write the definition of the overloaded operators that is needed for class Text, to allow the given `main` function work without any error. **Also, add function prototypes in the class definition, above.**

Notes: 1 mark will be deducted from each UNNECESSARY operator that you overload

~~[] >>~~

```
istream& operator >>(istream& is, const Text& t){
    ~~for(int i=0; i< lengthM; i++){~~
        is >> t.storageM[i];
    ~~}~~
    return is;
}
char& Text operator[](int index){
    return storageM[index];
}
```

0.75 5.25 2.5 1

Question 8 (7 marks) - Multiple Choice Questions

1. A static data member in a C++ class:
   A. Must be declared globally outside the class definition
   B. Will be initialized to zero automatically
   C. All of the above: A, B, and C are all correct
   D. None of the above are correct

2. What is the difference between a static member function and a non-static member function in a C++ class?
   A. A static member function can be called even without existence of a class object.
   B. A static member function doesn't have direct access to the class private data members
   C. A static member function does not have a 'this pointer'
   D. All of the above
   E. None of the above

3. Which one of the following statements are correct?
   A. An abstract class is a class that is at the root of the class hierarchy
   B. An abstract class is a class that cannot have an object
   C. A and B are both correct.
   D. None of the above is correct

4. Which one of the following statements is correct? Select the best answer.
   A. An abstract class is a class that contains a virtual function.
   B. An abstract class is a class that contains at least one pure virtual function.
   C. An abstract class is class that is derived from a virtual base class
   D. A, and B are correct answers
   E. All of the above are correct answers
   F. None of the above is a correct answer

5. Which one of the following statements is correct about a static member function in a C++ class? Select the best answer.
   A. A static member function can be called even without existence of a class object.
   B. A static member function doesn't have direct access to the class private data members
   C. A static member function does not have a 'this pointer'
   D. B and C are correct answers
   E. All of the above are correct answers
   F. None of the above

6. Which one of the following statements are correct?
   A. An abstract class is a class that is at the root of the entire class hierarchy
   B. An abstract class is a class that cannot have an object.
   C. An abstract class can be a reference or pointer argument of a global function
   D. B and C are both correct.
   E. All of the above are correct
   F. None of the above is correct

7. Which one of the following statements is a correct in C++?
   A. An overloaded operator function must be always a member of a class or a friend of a class
   B. An overloaded operator function must have at least a class objec as its argument.
   C. An overloaded operator cannot return a pointer, but can return a reference.
   D. A and B are correct answers
   E. All of the above are correct answers
   F. None of the above is a correct answer