National University of Computer and Emerging Sciences

FAST School of Computing **Summer-2023** **Islamabad Campus**

CS-1004: Object Oriented Programming (Solution)

Serial No:

Mid Term Exam

Total Time: 2 Hours

Total Marks: 80

Monday, 10th July, 2023

Course Instructors

Majid Hussain, Sami Ullah and Jawad Hassan

Signature of Invigilator

Student Name Roll No. Course Section Student Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **Fifteen (15)** different printed pages including this title page. There are **Three (3)** questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Total
Marks Obtained				
Total Marks	30	40	25	80

National University of Computer and Emerging Sciences

FAST School of Computing　　Summer-2023　　Islamabad Campus

Question 1 [3 + 3 + 3 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 4 + 6 + 4 = 30 Marks]

a) **(3 Marks)** Provide the output of the following program

```
int main(){
    int a = 22, *ptr = &a;
      char ch = 'G', &cho = ch;

      cho += a;
      cho -= 10;
      *ptr += ch;
      cout << a << ", " << ch << endl;
      return 0;
}
```

105, S

b) **(3 Marks)** Provide the output of the following program:

```
int main(){
      int num[5];
      int* p;
      p = num;
      *p = 10;
      p+=4;
      *p = 20;
      p = &num[2];
      *p = 30;
      p = num + 1;
      *p = 40;
      p = num;
      *(p + 3) = 50;
      for (int i = 0; i < 5; i++)
            cout << num[i] << ", ";
      return 0;
      }
```

10, 40, 30, 50, 20

c) **(3 Marks)** Assuming the address of ptr is 1000 and address of c[0] is 2000, what will be the output of the following code?

```
int main (){
      char *ptr, c[7] = "SUMMER";
      ptr = c+3;
      cout << *(ptr+2) << " " << ptr << " " << &ptr << " " << (*ptr) << " " << &c;
      }
```

R MER 100 M 2000

d) **(1 Mark)** What is correct about the static data member of a class?

A.	A static member function can access only static data members of a class.	B.	A static data member is shared among all the object of the class.
C.	A static data member can be accessed directly from main()	D.	Both A and B

e) **(1 Mark)** Which of the following is the correct declaration of a constant function?

A.	constintMyfun(void) { /* statements */ }	B.	int const Myfun(void) { /* statements */ }
C.	intMyfun(void) const { /* statements */ }	D.	Both **B** and **C**

f) **(1 Mark)** Which of the following is TRUE about static functions:

A.	Static functions can access non static data members	B.	Cannot be accessed by using class name with scope resolution operator
C.	No **this** pointer for **static** functions, they exist independent of objects.	D.	All A, B and C

g) **(1 Mark)** Which of the following statement is VALID about constant member functions:

A.	Member functions declared constant can modify their objects.	B.	Constructor and destructor can be constant.
C.	Non-constant objects can call constant member functions.	D.	None of above

h) **(1 Mark)** Which of the following statement is not VALID about Destructors:

A.	Called explicitly when scope of an object ends	B.	Can be overloaded
C.	Same name as class	D.	All A, B and C

i) **(1 Mark)** Which of the following statement is VALID about Friend functions of a class:

A.	Friend functions are defined within the scope of class	B.	Can access only public member of a class
C.	Friend functions are only external functions that appeared in to friendship granted list	D.	None of above

j) **(1 Mark)** General syntax of defining friend to a class is:

A.	*ReturnType friend FunctionName (ParameterTypeList);*	B.	*friend ReturnType FunctionName (ParameterTypeList);*
C.	*ReturnType FunctionName friend (ParameterTypeList);*	D.	None of above

k) **(4 Marks)** What is the output of the following program segment? Identify errors (if any).

```
class myclass {
private:
	int x;       //no Error in this line of code
	const int y; //no Error in this line of code
public:
	myclass(int x1 = 1, int y1 = 2):y(y1){
		x = x1;
		~~y = y1;~~
	}
	void setx(int x1){
		~~y = x1;~~
		x = x1;

	}
	void print() {
		cout << x << " " << y << endl;
	}
};
int main(){
	myclass q1;
	q1.setx(10);
	myclass q2(11, 12);
	q1.print();
	q2.print();
}
```

OUTPUT:

```
10  2
11  12
```

l) **(6 Marks)** What is the output of the following program segment? Identify errors (if any).

```
class yourclass {
private:
	int val;
public:
	yourclass(int i) {
		(*this).val = i;
		cout << " Con: " << val << " \n";
	}
	~yourclass(){
		cout << " Des: " << val << " \n";
	}
};
yourclass O1(1);
int build() {
	yourclass O2(4);
	static yourclass O3(5);
	return 0;
}
int main() {
	yourclass O4(2);
	static yourclass O5(3);
	build();
	yourclass O6(6);
	return 0;
}
```

OUTPUT:

```
Con: 1
Con: 2
Con: 3
Con: 4
Con: 5
Des: 4
Con: 6
Des: 6
Des: 2
Des: 5
Des: 3
Des: 1
```

m) **(4 Marks)** What is the output of the following program segment? Identify errors (if any).

```
class Rectangle
{
   private:
    int w;  int h;

   public:
     Rectangle () {}
    void SetWidth(int ww) { w=ww; }
    void SetHeight(int hh) { h=hh;}
    int getArea() { return w*h; }
};

int main() {
  Rectangle r1;
  Rectangle *ptr = &r1;
  Rectangle &ref = r1;
  Rectangle* &ref2 = ptr;

  r1.SetHeight(5);
  r1.SetWidth(4);
  cout<<"\n Area (object) = "<<r1.getArea();
  cout<<"\n Area (pointer) = "<<ptr->getArea();
  cout<<"\n Area (reference to obj) = "<<ref.getArea();
  cout<<"\n Area (reference to pointer) = "<<ref2->getArea();
  return 0;
}
```

Output:

```
Area (object) = 20
Area (pointer) = 20
Area (reference to obj) = 20
Area (reference to pointer) = 20
```

Question 2 [15 + 15+ 10 = 40 Marks]

a) (15 Marks) In architectural drawings, the distances are measured in feet and inches according to the English system of measurement. There are 12 inches in a foot. The length of a living room, for example, might be given as 15’–8”, meaning 15 feet plus 8 inches. The hyphen isn’t a negative sign; it merely separates the feet from the inches. Figure 1 shows typical length measurements in the English system. Suppose you want to create a drawing or architectural program that uses the English system, it will be convenient to store distances as two numbers, representing feet and inches. Develop a complete class with appropriate constructor and destructor as well as setter and getter functions to model and program the Distance representation in the English System. Also, include two functions, getDistance to get input from the user and showDistance to display the distance in the prescribed format. We need you to keep track of how many instances of the class are created in main or any other function. The class should define overloaded addition (+), less than (<), addition assignment (+=) operator.

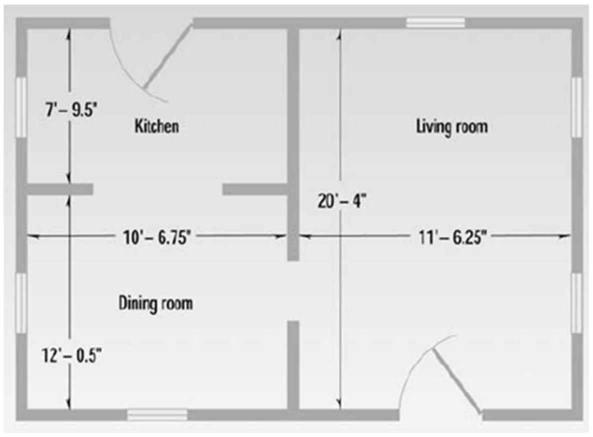

Figure 1: English System of Measurement

```
#include <iostream>
using namespace std;

class Distance {
private:
    int feet;
    int inches;
    static int instanceCount; // Static variable to track the number of instances

public:
    // Constructor
    Distance(int ft = 0, int in = 0) : feet(ft), inches(in) {
        instanceCount++;
    }

    // Destructor
    ~Distance() {
        instanceCount--;
    }

    // Setter function to set feet and inches
    void setDistance(int ft, int in) {
        feet = ft;
        inches = in;
    }

    // Getter function to get feet and inches
    void getDistance() {
        cout << "Enter feet: ";
        cin >> feet;
        cout << "Enter inches: ";
        cin >> inches;
    }

    // Display distance in the prescribed format
    void showDistance() {
        cout << feet << "’-" << inches << "”" << endl;
    }

    // Static function to get the number of instances
    static int getInstanceCount() {
        return instanceCount;
    }
};

int Distance::instanceCount = 0; // Initialize the static variable
int main() {
    // Create instances of the Distance class
    Distance d1(15, 8);
    Distance d2;

    // Display distances
    cout << "Distance 1: ";
    d1.showDistance();

    cout << "Distance 2: ";
    d2.getDistance();

    cout << "Distance 2: ";
    d2.showDistance();

    // Get the number of instances
    cout << "Number of instances created: " << Distance::getInstanceCount() << endl;

    return 0;
}
```

b) (15 Marks) Implement a class Book that contains a book's author, title and price as its data members. Make appropriate constructor, destructor, setter, getter and other utility function (if required).

Implement following global functions that perform following functions on class objects:

1) Comparison of two books should be done using a function with the following prototype:
   int compare(book *a, book *b);
   The function should return a negative value if book a comes before b, and a positive value if b comes before a (You may use the function Compare (str1, str2) to compare two strings). Zero should be returned in case of an exact match.

2) Swapping of two book structures should be done using a function with the following prototype:
   void swap(book *a, book *b);

3) Sorting of array of type Book:
   void sort (book *array, int count);

Hint: Sort() function may use **swap**() and **compare**() functions for performing soring of array of books**.**

You have complete freedom to use or implement any supporting functions as per your requirement with proper justification.

```
#include <iostream>
#include <string>
using namespace std;

class Book {
private:
   string author;
   string title;
   double price;

public:
   // Constructor
   Book(const string& auth, const string& t, double p) : author(auth), title(t), price(p) {}

   // Destructor (optional, not explicitly required in this case study)
   ~Book() {}

   // Getter functions
   string getAuthor() const { return author; }
   string getTitle() const { return title; }
   double getPrice() const { return price; }

   // Setter functions (optional, not explicitly required in this case study)
   void setAuthor(const string& auth) { author = auth; }
   void setTitle(const string& t) { title = t; }
   void setPrice(double p) { price = p; }
};
```

```
// Comparison function for books
int compare(const Book* a, const Book* b) {
    // You may use any string comparison function here
    return a->getTitle().compare(b->getTitle());
}

// Swapping function for books
void swap(Book* a, Book* b) {
    Book temp = *a;
    *a = *b;
    *b = temp;
}

// Sorting function for array of books
void sort(Book* array, int count) {
    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (compare(&array[j], &array[j + 1]) > 0) {
                swap(&array[j], &array[j + 1]);
            }
        }
    }
}

int main() {
    // Creating an array of Book objects
    Book books[] = {
        Book("Author C", "Title C", 12.5),
        Book("Author A", "Title A", 10.0),
        Book("Author B", "Title B", 15.75)
    };

    int bookCount = sizeof(books) / sizeof(books[0]);

    // Sorting the array of books based on titles
    sort(books, bookCount);

    // Displaying the sorted array of books
    cout << "Sorted Books:\n";
    for (int i = 0; i < bookCount; i++) {
        cout << "Title: " << books[i].getTitle() << ", Author: " << books[i].getAuthor() << ", Price: $"
<< books[i].getPrice() << endl;
    }

    return 0;
}
```

c) (10 Marks) Write recursive function(s), which find(s) substring in a string. No loops allowed whatsoever, and you can write maximum two recursive functions apart from main function.

Example:
1. Input: str = "OOP Summer 2023"
substr = "mer"
Output: true
2. Input: str = " OOP Fall 2023"
substr = "Final"
Output: false
Function Prototype: bool findSubString(char *str , char *substr);
Note: Don't use Static variable(s).

```
#include <iostream>
using namespace std;

bool compareSubString(const char* str, const char* substr) {
   // Base case: If both strings reach the end simultaneously, it means the substring is found.
   if (*str == '\0' && *substr == '\0') {
      return true;
   }

   // If either of the strings reaches the end while the other has not, it means the substring is not found.
   if (*str == '\0' || *substr == '\0') {
      return false;
   }

   // If the current characters of both strings match, check the remaining characters.
   if (*str == *substr) {
      return compareSubString(str + 1, substr + 1);
   }

   // If the characters do not match, start comparing the substring from the beginning with the next
character in the string.
   return compareSubString(str + 1, substr);
}

bool findSubString(const char* str, const char* substr) {
   // Base case: If the substring is an empty string, it is always found.
   if (*substr == '\0') {
      return true;
   }

   // If the string is an empty string and the substring is not, it is not found.
   if (*str == '\0') {
      return false;
```

```
    }

    // Check if the current character of the string is the same as the first character of the substring.
    if (*str == *substr) {
        // If it matches, check the rest of the string for the substring.
        if (compareSubString(str, substr)) {
            return true;
        }
    }

    // Move to the next character in the string and continue searching for the substring.
    return findSubString(str + 1, substr);
}

int main() {
    const char* str1 = "OOP Summer 2023";
    const char* substr1 = "mer";
    bool result1 = findSubString(str1, substr1);
    cout << "Output 1: " << (result1 ? "true" : "false") << endl;

    const char* str2 = "OOP Fall 2023";
    const char* substr2 = "Final";
    bool result2 = findSubString(str2, substr2);
    cout << "Output 2: " << (result2 ? "true" : "false") << endl;

    return 0;
```

Question 3 [5 + 5 + 5 + 10 = 25 Marks]

a) (5 Marks) For the code given below, do the following:

1. Identify errors
2. Correct errors
3. Make sure there is no shallow copy
4. Write the output

```
class myClass{
private:
        char*  x;
        int   y;
        const int z;

public:
        static int a;
        myClass(char* = NULL, int = 0, int = 0);
        myClass(const myClass&);
        ~myClass();
        static int getA() {
                y++;
                return a;
        }
};

myClass::myClass(char* xPtr, int yVal, int zVal)
{
        x = xPtr; y
        = yVal; z
        = zVal;
        ++a;
}
myClass::myClass(const myClass& c)
{
        x  = c.x;
        y  = c.y;
        z  = c.z;
}
myClass::~myClass() {
        --a;
}
int main()
{
        cout << myClass::a<<endl;                       //output:
        myClass c1, c2;
```

```
    myClass c3(c1);
    cout << c1.a << endl;                    //output:

    if (c1.a == c2.a) {
        char arr[] = "new string";
        myClass c4(arr, 5, 6);

        cout<< myClass::getA()<<endl;        //output:
    }

    cout<<c3.getA()<<endl;                   //output:

    return 0;
}
```

b) (5 Marks) What would be the output produced by executing the following C++ code? Explain the error(s) or bug(s) if there is/are any.

```
#include<iostream>
using namespace std;
class A {
 int x;
public:
A (int val = 0) : x(val) {cout << "A " << x << endl;}
A (const A& a) {
 x = a.x - 1;
 cout << "B " << x << endl;
 }
 void operator= (const A& a){
  x = a.x - 2;
  cout << "D " << x << endl;
 }
 void SetX (int x) {this->x = x;}
 int GetX() { return this->x;}
};
void Out(A a) {
 cout << "Out "<<a.GetX() << endl;
}
A In(A &x) {
 cout<<"Ref A "<<x.GetX()<<endl;
 x.SetX(x.GetX() + 4);
 return x;
}
int main() {
 A a(1), b=In(a);
 Out(b);
 return 0;
}
```

c) (5 Marks) What would be the output produced by executing the following C++ code? Explain the error(s) or bug(s) if there is/are any.

```
#include <iostream>
#include <string>
using namespace std;
class Point{
int x, y;
public:
Point(int x=0, int y=0):x(x),y(y){}
Point(const Point  p):x(p.x),y(p.y){
cout<<"The copied point";
cout<<" ("<<x<<","<<y<<") " <<endl;
    }
string getPoint() {return "("+to_string(x)+","+to_string(y)+")";}
~Point(){cout<<"Point is destroyed"<<endl;}
};
class Circle{
Point center;
float radius;
public:
Circle():center(0,0),radius(0){}
Circle(Point p):center(p){}
Circle(const Circle  c):center(c.center), radius(c.radius){
cout<<"The copied circle "<<center.getPoint()<<endl;
    }
~Circle(){cout<<"Circle is destroyed"<<endl;}
};
int main() {
Point p2;
Circle c2(p1);
Circle c3(c2);
return 0;
}
```

d) (10 marks) Implement a class Sequence to store a sequence of non-negative integer values, and the length of the sequence. The class has the following private data members:
1**. int length –** the length of the sequence
2**. int *pseq –** a pointer to a dynamic integer array holding sequence of integers
The class shall provide the following public methods:
1. **Sequence() –** a default constructor that initializes length to 10 and store the sequence of all zeros in an array.
2. **Sequence(int lengthVal, int n1=0,int n2=0,int n3=0, int n4=0, int n5=0, int n6=0, int n7=0,**
int n8=0, int n9=0, int n10=0) – another parameterized constructor should initialize the length and array to sequence values passed in the arguments.
3. **Sequence(Sequence &s) –** a copy constructor that creates a copy of a Sequence object.
4. **int getLength() –** a getter for length.
5**. int* getSeq() –** a getter for the sequence of numbers
6. **void Sort(int n) –** a function that sorts the first n elements in the sequence array. You cannot use Bubble Sort Algorithm
7. **int RemoveDuplicates() –** a function that removes duplicates from the sequence in-place (without creating another array). Since you cannot change the size of the array or create a new one, you must place the result in the first part of the array. If there are n unique elements in the array, then the first n elements of array should hold the final result sorted in ascending order. It does not matter what you leave beyond the first n elements in the array. Return n after placing the final result in the first k elements of the array.
Return n after placing the final result in the first k elements of the array.
Example
Seq array = {3,1,2,2,1,7}
n = 4
Sequence array after duplicate removal = {1,2,3,7,dk,dk}
dk = don’t care about values here

8**. void Rotate(int steps) –** a method that rotates the sequence elements clockwise for the given steps
Example
Seq array = {1,3,4,6,2,6,0}
steps = 3
Sequence array after rotation = {2,6,0,1,3,4,6}

9. **~Sequence() –** a destructor to deallocate the dynamically created array.