# Indian Institute of Technology Kharagpur

## Class Test I 2022-23

**Date of Examination:** **Jan, 2023** **Duration:** **45 Minutes**

**Subject No.:** **CS20006/CS20202** **Subject:** **Software Engineering**

**Department/Center/School:** **Computer Science** **Credits:** **3** **Full marks:** **20**

**Name:** ______________________________________________

**Roll Number:** ______________________

### Instructions

i. **Please write your name and roll number above before attempting any solution**.

ii. Write your answers in this question paper itself. It has been given a booklet form for this purpose.

iii. Use of electronic calculators only is permitted. No extra resources viz. graph papers, log-tables, trigonometric tables would be required.

iv. **All questions are compulsory.** Be brief and precise. Mysterious or unsupported answers will not receive full marks.

v. **A few extra blank sheets are provided at the end**. Please use them, if for any question, you need extra space.

| Question: | 1 | 2 | 3 | Total |
|---|---|---|---|---|
| Points: | 5 | 5 | 10 | 20 |
| Score: | | | | |

1. (a) (1 point) Which among the following data structures is commonly used to convert an infix expression to a postfix expression: (a) array (b) stack (c) queue?

**Solution:** stack

(b) (1 point) Write the statement to declare **fp** as a pointer to a file in C.

**Solution:** FILE *fp;

(c) (1 point) Consider the following code segment.

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

int main() {
    bool i = true, j = false, k = false;
    cout << (i || j && k);
    return 0;
}
```

What will be printed in the console?

**Solution:** 1

(d) (1 point) Consider the following program.

```
1  #include <iostream>
2  #include <string>
3  #include <cstring>
4  using namespace std;
5
6  int main() {
7      string greet = "Hello Student";
8      _______________;
9      cout << greet;
10     return 0;
11 }
```

Fill in the blank at line 8 such that the output is **Hello**.

**Solution:** greet.resize(5)

(e) (1 point) Consider the following code segment.

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

int divide(int a, int b){
    return a/b;
}

int main(){
    cout << divide(015,2) << endl;
}
```

What will be printed in the console?

**Solution:** 6. (015 is 15 in octal notation.)

2. (a) (2 points) Consider the following code segment.

```
1  #include <iostream>
2  using namespace std;
3
4  void increment(int x1, ________, ________){
5      x2 = ++x1;
6      *x3 = x1++;
7  }
8
9  int main(){
10     int a = 3, b, c;
11     increment(a,b,&c);
12     cout << a << " " << b << " " << c;
13     return 0;
14 }
```

What should be the last two formal parameters in the parameter list denoted by two dashed lines at line 4 such that the output is `3 4 4`. Briefly explain your answer.

**Solution:** `int &x2, int *x3`
Since the changes made in `x2, *x3` in function `increment()` need to be reflected in the variables `b` and `c` in `main()`, these are either pass-by-reference or pass-by-address. From the function calling point, it can be observed that `b` is passed-by-reference and `c` is passed-by-address. Thus, the header of `increment()` function must be:
`void increment(int x1, int &x2, int *x3)`

(b) (3 points) Consider the following code segment.

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

void func1(string & a, int & b, int c){
    a += "!";
    b--;
    c = c + 5;
    cout << a << " " << b << " " << c << endl;
}

int main() {
    string a = "IIT Kharagpur"; int b = 4, c = 6;

    func1(a, b, c);
    func1(a, b, b);

    cout << a << " " << b << " " << c << endl;

    return 0;
}
```

List below the output produced by this program.

**Solution:**
```
IIT Kharagpur!  3 11
IIT Kharagpur!!  2 8
IIT Kharagpur!!  2 6
```

3. (10 points) The roots of the quadratic equation $ax^2+bx+c = 0$ may be real distinct (represented as a **struct** of the defined type **realDTyp** with members **r1** and **r2**), complex conjugate (represented as a **struct** of the defined type **cplxCTyp** with members **r** and **s**) or real repeated (represented as a single **float** variable **rr**).

The type of the root is identified by an **enum** named **qeRootKey** of defined type **qeRootKeyTyp** with elements **RealD**, **RealR** and **CplxC**. The root values are to be stored in a variable **qeRootVal** (represented as a **union** of the defined type **qeRootValType** with members **rootRD** of type **realDTyp**, **rootCC** of type **cplxCTyp**) and **rootRR** of type **float**).

The roots are stored in a **struct** of the defined type **qeRootVarietyTyp** with members **qeRootKey** of type **qeRootKeyTyp** and **qeRootVal** of type **qeRootValTyp**.

The prototype of a function **quadEqnRoots()** is to be written to take the coefficients of the quadratic equation (represented as a **struct** of the defined type **quadCoeffTyp** with members **c0** for $c$, **c1** for $b$ and **c2** for $a$). The function should return the roots as a **struct**, as described above. Fill up the blank spaces to complete the above type definitions and the prototype for **quadEqnRoots()**.

```
#include <stdio.h>
int main(){
    typedef struct{ // real distinct
        ________________________________
    } realDTyp;
    typedef struct{ // complex conjugate
        ________________________________
    } cplxCTyp;
    float rr;

    typedef ____________________{
        ________________________________
    } qeRootKeyTyp;
    typedef union{
        ________________________________
        ________________________________
        ________________________________
    }qeRootValTyp;
    typedef struct{
        ________________________________
        ________________________________
    }qeRootVarietyTyp;
    typedef struct{
        ________________________________
    }quadCoeffTyp;

    ____________________ quadEqnRoots(________________________________);

    return 0;
}
```

**Solution:**

```
#include <stdio.h>
int main(){
    typedef struct{ // real distinct
        float r1, r2;
    } realDTyp;

    typedef struct{ // complex conjugate
        float r, s;
    } cplxCTyp;

    float rr;

    typedef enum qeRootKey{
        RealD, RealR, CplxC
    } qeRootKeyTyp;

    typedef union{
        realDTyp rootRD;
        cplxCTyp rootCC;
        float rootRR;
    }qeRootValTyp;

    typedef struct{
        qeRootKeyTyp qeRootKey;
        qeRootValTyp qeRootVal;
    }qeRootVarietyTyp;

    typedef struct{
        float c0, c1, c2;
    }quadCoeffTyp;

    qeRootVarietyTyp quadEqnRoots(quadCoeffTyp coeffs);

    return 0;
}
```