A23 CS101 Endsem Answers and Rubrics HELLO

Q1

This is the correct code for ve rsion 1

```
for (unsigned int i = 0; i < N; i++) { // Line 1
//do not deduct marks if i is changed to 1 in Line 1
//If student changes i to I in line 1 then he should be given
//1 mark total of line 2 and 3. Check Line 2 and 3 accordingly.
    const int current = data[i]; // Line 2          Marks=0.5
    unsigned int newPosition = i;// Line 3          Marks=0.5

while (newPosition > 0 && current < data[newPosition - 1]) //line4
                                                    Marks=3
{
        // Shift elements to the right
data[newPosition] = data[newPosition - 1];//Blank A  Marks=2
        newPosition--;// Line 6                      Marks=0.5
    }

    // Insert the current element into the correct position
    data[newPosition] = current;                     // Blank B
    Alternate Solution : data[newPosition] = data[i]; // Blank B
                                                    Marks=2
}
```

This is the correct code for version 2

```
for (unsigned int i = 1; i < N; i++) {
    for  (unsigned int j = i; j > 0; j- - ) { // Blank A        Marks=4.5
        if (data[j] < data[j-1]) { // Blank B                  Marks=2
            // Swap elements
            const int temp = data[j];
```

```
            data[j] = data[j-1];
            data[j-1] = temp;
        }
    }}
```

Q2

(a)

```
#include <iostream>
using namespace std;
struct student {
      unsigned int RollNum;
      unsigned int BestFriendsCount;
      student * BestFriends[3];
};

bool Search (student & Y, student & X, bool * Checked)
{
      Checked[Y.RollNum] =true; // or 1                    0.5 marks
      cout << "checking: " << Y.RollNum << endl;
      if (Y.RollNum == X.RollNum)
      {
            return true; // or 1                           0.5 marks
      }
      bool found = false;
      for (unsigned int k = 0; k < Y.BestFriendsCount; k++)
      {?
            student Z = *(Y.BestFriends[k]);
            if (!Checked[Z.RollNum]) /.?
                // or if (Checked[Z.RollNum] == false)   1 mark
                  found = Search (Z,X,Checked);          0.5, 0.5
            marks
            if (found)                                  1 mark
                  return true; or found; or 1;            0.5 marks

      }
      return false; or found; or 0;                      0.5 marks
}
```

(b) – For the output, 1 mark for each correct line. If some line is wrong, do not check the ones after that

```
checking: 0;
checking: 1
checking: 3
checking: 5
checking: 4
```

```
Verbose:
TODO: Rubrics for execution trace will be finalized after looking at
students' answers
---
0     4
Call: Y.RollNum = A, X.RollNum = E
checking: 0
Call: Y.RollNum = B, X.RollNum = E
checking: 1
Call: Y.RollNum = D, X.RollNum = E
checking: 3
Call: Y.RollNum = F, X.RollNum = E
checking: 5
Call: Y.RollNum = E, X.RollNum = E
checking: 4
1
```

(c) – If some line is wrong, do not check the ones after that

```
checking: 0   0.7 marks
checking: 1   0.7 marks
checking: 3   0.7 marks
checking: 5   0.7 marks
checking: 4   0.7 marks
checking: 2   0.7 marks
checking: 6   0.8 marks
Verbose:
---
0     7
Call: Y.RollNum = A, X.RollNum = H
checking: 0
Call: Y.RollNum = B, X.RollNum = H
checking: 1
Call: Y.RollNum = D, X.RollNum = H
checking: 3
Call: Y.RollNum = F, X.RollNum = H
```

```
checking: 5
Call: Y.RollNum = E, X.RollNum = H
checking: 4
Call: Y.RollNum = C, X.RollNum = H
checking: 2
Call: Y.RollNum = G, X.RollNum = H
checking: 6
0
```

Q3

(a)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

In the context of a regression problem, the goal is to find the parameters a, b, and c in the quadratic model $f(x) = ax^2 + bx + c$ that best fit the given data points (x_i, y_i) for $i = 0, 1, \ldots, 100$.

A common quality-of-fit criterion in regression is the sum of squared errors (SSE). The SSE is defined as the sum of the squared differences between the predicted values ($f(x_i)$) and the actual observed values (y_i) for all data points. Mathematically, the SSE is given by:

$$SSE = \sum_{i=0}^{100}(f(x_i) - y_i)^2$$

Substitute the expression for $f(x)$ into the SSE formula:

$$SSE = \sum_{i=0}^{100}(a(x_i)^2 + b(x_i) + c - y_i)^2$$

Marking Scheme for I)- BOTH SSE and MSE are correct. The student has to write any one. Award 3 marks if the mathematical expression is fully correct and 0 marks if the expression is I .incorrect in any way.

II) *Marking Scheme-*

In PART 3 and 4, assign 1.5 marks for each step; award 0.5 marks if partially correct.

In PART 5, allocate 1 mark for a correct answer and 0.5 marks if the contents of matrices are partially correct.

II. Derive Equations in the Form of $Mv = b$:

1. **Model Prediction:**

$$f(x_i) = ax_i^2 + bx_i + c$$

2. **Loss Function (Mean Squared Error):**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)^2$$

3. **Partial Derivatives with Respect to a, b, and c:**

$$\frac{\partial \text{MSE}}{\partial a} = \frac{2}{n}\sum_{i=1}^{n} x_i^2(ax_i^2 + bx_i + c - y_i)$$

$$\frac{\partial \text{MSE}}{\partial b} = \frac{2}{n}\sum_{i=1}^{n} x_i(ax_i^2 + bx_i + c - y_i)$$

$$\frac{\partial \text{MSE}}{\partial c} = \frac{2}{n}\sum_{i=1}^{n}(ax_i^2 + bx_i + c - y_i)$$

4. **Set Partial Derivatives to Zero and Solve for a, b, and c:**

$$\frac{\partial \text{MSE}}{\partial a} = 0 \quad \Rightarrow \quad a\sum_{i=1}^{n} x_i^4 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i^2 y_i$$

$$\frac{\partial \text{MSE}}{\partial b} = 0 \quad \Rightarrow \quad a\sum_{i=1}^{n} x_i^3 + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i$$

$$\frac{\partial \text{MSE}}{\partial c} = 0 \quad \Rightarrow \quad a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i + nc = \sum_{i=1}^{n} y_i$$

5. **Matrix Form $Mv = b$:**

$$M = \begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & n \end{bmatrix}, \quad v = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad b = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}$$

III)Method: Gauss Jordan Elimination/ Least Square along with its description.

Marking Scheme- 2 Marks if proper description of algorithm is given 0.5 if not.

Q4

```
#include <iostream>
#include<cstdlib>
//If written 'Not Required' or written any valid library like cstdlib marks will be provided            Marks=1
using namespace std;
int main () {
      rand(-1); //Write srand(-1) //srand(0) or remove the line. Unnecessary or remove -1 Marks=1
      long int N = 1e3  or N = 0;                                   Marks=1
      double average = 0;
      while (N --> 0)  or while(N++<0) if N is set to zero
      {`
            const double temp = rand()*1.0/RAND_MAX;                Marks=2

            ;...        OR
            const double temp = (double)rand()/RAND_MAX;
                        OR
            const double temp = rand()/(1.0+RAND_MAX);
            unsigned char draw = 0; //If char is changed to int/float no negative marks
            if ( temp<=1.0/21) draw = 1;                    Marks=1.5
            if (temp>1.0/21 && temp<=3.0/21) draw = 2;      Marks=1.5
            if (temp>3.0/21 && temp<=6.0/21) draw = 3;      Marks=1.5
            if (temp>6.0/21 && temp<=10.0/21) draw = 4;     Marks=1.5
            if (temp>10.0/21 && temp<=15.0/21) draw = 5;    Marks=1.5
            if(temp>15.0/21) draw = 6;                      Marks=1.5

                              OR

            if ( temp*21 < 1) draw = 1;
```

```
            if ( temp*21 >= 1 && temp*21< 3) draw = 2;
            if ( temp*21 >= 3 && temp*21< 6) draw = 3;
            if ( temp*21 >= 6 && temp*21< 10) draw = 4;
            if ( temp*21 >= 10 && temp*21< 15) draw = 5;`
            if ( temp*21 >= 15 ) draw = 6;

            average = average + draw;
      }
      average = average / 1000;  or 1e3 or average/N if N is set to zero
            Marks=1
      cout.precision (10);
      cout << average << endl;
}
```

\

Q5

```
#include <iostream>
using namespace std;
template <class T>
class Vector
{
private:
    unsigned int size;
    T *data;
    void copy(const Vector<T> &);

public:
    Vector(unsigned int n = 8) : size(n), data(new T[size])
    {
        if (n == 0)
            data = NULL;
    }

    Vector(const Vector<T> &v) : size(v.size), data(new T[size])
    {
        copy(v);
    }
```

```
    ~Vector()
    {
        ////////////added this line/////////////
        delete[] data;                                          Marks = 2
    } // 1 and only 1 statement

    unsigned int length() const
    {
        return size;
    }

    T &operator[](unsigned int i) const
    {
        // return data[i / size];
        return data[i];                                          Marks =
2
    }
    Vector<T> &operator=(const Vector<T> &) const; //remove const
Marks=1.5
};

template <class T>
void Vector<T>::copy(const Vector<T> &v)
{
    const unsigned int num = (size < v.size) ? size : v.size;
    for (unsigned int i = 0; i < num; i++)
        data[i] = v.data[i]; // inverted                          Marks =
2
}

template <class T>
Vector<T> &Vector<T>::operator=(const Vector<T> &v) const
Marks=1.5
                                                    // remove const
{
    // 1 and only 1 statement
    delete[] data;                                               Marks =
2
    data = new T[v.size];
    size = v.size; ////inverted                                  Marks = 2
    copy(v);
    // 1 and only 1 statement
    return *this;                                                Marks = 2
}
```

```
int main()
{
    Vector<int> v(10);
    for (int i = 0; i < v.length(); i++)
        v[i] = i;
    for (int i = 0; i < v.length(); i++)
        cout << v[i] << " ";
    cout << endl;
    cout << "--------------------" << endl;
    Vector<int> w(v);
    for (int i = 0; i < w.length(); i++)
        cout << w[i] << " ";
    cout << endl;
    cout << "--------------------" << endl;
    Vector<int> x;
    for (int i = 0; i < x.length(); i++)
        x[i] = -i;
    for (int i = 0; i < x.length(); i++)
        cout << x[i] << " ";
    cout << endl;
    cout << "--------------------" << endl;
    v = w = x;
    for (int i = 0; i < v.length(); i++)
        cout << v[i] << " ";
    cout << endl;
    for (int i = 0; i < w.length(); i++)
        cout << w[i] << " ";
    cout << endl;
    cout << "--------------------" << endl;
    Vector<float> y(3);
    for (int i = 0; i < y.length(); i++)
        cout << y[i] << " ";
    cout << endl;
    Vector<float> z(0);
    for (int i = 0; i < z.length(); i++)
        cout << z[i] << " ";
    cout << endl;
    z = y;
    for (int i = 0; i < z.length(); i++)
        cout << z[i] << " ";
    cout << endl;
}
```