Université INUKA

Rapport final web mobile Déc. 2023

Calculatrice Scientifique

Liste des membres de ce groupe :

- Paul Denis COQUILLON 33466
- Jean Roodsen CENAT 33390
- Schneider B.M Maxena 33818
- Ada B. Francois 33888
- Narky TURENNE 33908

Prof : Foko sindjoung miguel landry

Date : 20 Décembre 2023

- **Sc. Calculator :** est une application d'Android codé avec Java pour le back end et XML pour le design.

❖ Partie Font End (XML)

- Pour que le graphique apparaisse comme vous le voyez ci-dessus on a créé et utilisé plusieurs d'autres modules XML

  Pour la fenêtre principale : activity_main.xml

  - **Architecture**

    LinearLayout

    TextView

    EditText

    Button

On a utilisé ce TextView pour l'écran secondaire placé au-dessus

```
<TextView
    android:id="@+id/ecran_2"
    android:layout_width="354dp"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#FFFFFF"
    android:cursorVisible="true"
    android:ellipsize="middle"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:gravity="bottom"
    android:maxLines="100"
    android:padding="10dp"
    android:scrollHorizontally="true"
    android:singleLine="true"
    android:text="0"
    android:textAlignment="viewEnd"
    android:textColor="#000000"
    android:textSize="30dp" />
```

On a utilisé cet EditText pour l'écran principal placé au-dessous du TextView afin d'afficher et d'éditer les calculs.

```
<EditText
    android:id="@+id/ecran_main"
    android:layout_width="354dp"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#FFFFFF"
    android:digits="0123456789"
```

```
android:ellipsize="middle"
android:focusableInTouchMode="true"
android:gravity="bottom"
android:maxLines="1"
android:padding="10dp"
android:scrollHorizontally="true"
android:singleLine="true"
android:textAlignment="viewEnd"
android:textColor="#000000"
android:textSize="40dp" />
```

< LinearLayout >

Pour cette LinearLayout principale, on a utilisé `android:weightSum="7"` pour indiquer qu'on va avoir 7 rangés d'éléments en LinearLayout.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:orientation="vertical"
    android:weightSum="7">
```

Pour cette LinearLayout , on a utilisé `android:weightSum="5"` pour indiquer qu'on va avoir 5 rangés de Butons bien aligné.

```
<!-- Ligne 2 -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="5">
```

Button

```
<Button
    android:id="@+id/btn_sin"
    style="@style/Button_fonction"
    android:background="@drawable/fonction"
    android:text="sin" />

<Button
    android:id="@+id/btn_7"
    style="@style/Button_chiffre"
    android:background="@drawable/chiffre"
    android:onClick="onButtonClick"
    android:text="7" />
```

Pour mieux gérer gérer l'interface des buttons vous pouvez remarquer qu'on a un style et un @drawable

Chaque style contient un nom différent :

- `style="@style/Button_fonction"`
- `style="@style/Button_chiffre"`

Et 2 @drawable différent :

- android:background="@drawable/fonction"
- android:background="@drawable/chiffre"

Style : Dans le dossier « themes » -> themes.xml

- Le style va définir la dimension et propriété de chaque bouton placé dans notre interface du projet

- **Chiffre**

```
<style name="Button_chiffre">
    <item name="android:layout_width">73dp</item>
    <item name="android:layout_height">60dp</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">20dp</item>
    <item name="android:padding">10dp</item>
    <item name="android:layout_margin">10dp</item>
    <item name="android:textColor">#00ACC1</item>
    <item name="background">@drawable/chiffre</item>
</style>
```

- **Fonction**

```
<style name="Button_fonction">
    <item name="android:layout_width">60dp</item>
    <item name="android:layout_height">60dp</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">16dp</item>
    <item name="android:textAllCaps">false</item>
    <item name="background">@drawable/fonction</item>
    <item name="android:padding">5dp</item>
    <item name="android:layout_margin">5dp</item>
    <item name="android:textColor">@color/white</item>
</style>
```

@drawable : Dans le dossier « drawable » -> chiffre.xml et fonction.xml

- Chiffre.xml va définir la dimension et propriété de chaque bouton chiffre de notre projet grâce à la propriété **shape** transformé en **ovale (oval)** accompagné de la balise **solide** afin de déterminer la couleur du shape.

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="oval">
    <solid android:color="@color/white"></solid>
</shape>
```

- fonction.xml va définir la dimension et propriété de chaque bouton fonction de notre projet

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="oval">
    <solid android:color="#69B5B5"></solid>
</shape>
```

MainActivity.java

C'est la partie la plus importante de notre projet, c'est dans cette partie qu'on a eu les conditions de fonctionnement de notre calculatrice.

- Initialisation des buttons

```
//Initialiser nos boutons
Button btn_ac, btn_p1, btn_p2, btn_del, btn_sin, btn_cos, btn_tan,
btn_ln, btn_log, btn_fact, btn_pow, btn_racine, btn_inv, btn_div,
btn_mult, btn_moins, btn_plus, btn_pow_x, btn_egale;
Button btn_7, btn_8, btn_9, btn_4, btn_5, btn_6, btn_3, btn_2, btn_1,
btn_zero, btn_point, btn_rd_deg;
```

- TextView

```
//TextView
static private TextView ecran_2;  static private EditText ecran_main;
```

- Autres

```
private String buton_text; private int cursorPosition;
DecimalFormat decimalFormat = new DecimalFormat("#.######");
DecimalFormat scientificFormat = new DecimalFormat("0.###E0");
private double baseForPowX = 0.0; private boolean waitingForExponent =
false;
// Initialisez l'état actuel (radians)
private static boolean isRadiansMode = true;
private BigInteger facto = BigInteger.ONE;
```

- Associer les variables à leurs Id

```
btn_ac = findViewById(R.id.btn_ac); btn_p1 = findViewById(R.id.btn_p1);
btn_p2 = findViewById(R.id.btn_p2); btn_del =
findViewById(R.id.btn_del);
//-----------------------------------
btn_sin = findViewById(R.id.btn_sin); btn_cos =
findViewById(R.id.btn_cos); btn_tan = findViewById(R.id.btn_tan);
btn_ln = findViewById(R.id.btn_ln); btn_log =
findViewById(R.id.btn_log);
//-----------------------------------
btn_fact = findViewById(R.id.btn_fact); btn_pow =
findViewById(R.id.btn_pow); btn_pow_x = findViewById(R.id.btn_pow_x);
btn_racine = findViewById(R.id.btn_racine);
btn_inv = findViewById(R.id.btn_inv); btn_div =
findViewById(R.id.btn_div);
//-----------------------------------
btn_mult = findViewById(R.id.btn_mult); btn_moins =
findViewById(R.id.btn_moins); btn_plus = findViewById(R.id.btn_plus);
btn_egale = findViewById(R.id.btn_egale);

//===================================
btn_7 = findViewById(R.id.btn_7); btn_8 = findViewById(R.id.btn_8);
btn_9 = findViewById(R.id.btn_9); btn_4 = findViewById(R.id.btn_4);
btn_5 = findViewById(R.id.btn_5);
//-----------------------------------
btn_6 = findViewById(R.id.btn_6); btn_3 = findViewById(R.id.btn_3);
btn_2 = findViewById(R.id.btn_2); btn_1 = findViewById(R.id.btn_1);
btn_zero = findViewById(R.id.btn_zero); btn_point =
findViewById(R.id.btn_point); btn_rd_deg =
```

```
findViewById(R.id.btn_rd_deg);
//-----------------------------------

ecran_main = findViewById(R.id.ecran_main);
ecran_main.setInputType(InputType.TYPE_CLASS_NUMBER);
ecran_2 = findViewById(R.id.ecran_2);
```

- Un event sur l'ecran principale afin de savoir la position des chiffres et le curseur :

```
ecran_main.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {
        // Ne rien faire avant le changement de texte
    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
        // Mettez à jour la position du curseur lors du changement
de texte
        cursorPosition = i + i2;
    }

    @Override
    public void afterTextChanged(Editable editable) {
        // Obtenez la longueur totale du texte après la modification
        int newCursorPosition = editable.length();
        // Déplacez le curseur à la droite après le changement de
texte
        ecran_main.setSelection(newCursorPosition);
    }
});
```

Cette partie nous aide à bien gérer la position du curseur à chaque clique, afin de mettre à jour la position du curseur lors du changement de texte, après une modification on peut obtenir la longueur totale du texte et après déplacez le curseur à la droite après le changement de texte.

```
//======== Onclick Listeners ==============
```

Button AC : Permet de supprimer entièrement une opération et un nombre de l'écran 2 et le principal.

```
btn_ac.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

Ce code modifie le texte de l'écran principal en noir.

ecran_main.setTextColor(getResources().getColor(android.R.color.black
));
        ecran_main.setText("");
        ecran_2.setText("0");
    }
});
```

Button Del : Permet de supprimer un seul chiffre de l'écran principal, vous pouvez cliquer le chiffre que vous voulez supprimer grâce au curseur.

```
btn_del.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Récupérer la position du curseur
        int cursorPosition = ecran_main.getSelectionStart();

        // Récupérer le texte actuel de l'EditText
        String valeur = ecran_main.getText().toString();

        // Vérifier si la position du curseur n'est pas à la première
position
        if (cursorPosition > 0) {
            // Créer une StringBuilder avec la valeur actuelle
            StringBuilder newValue = new StringBuilder(valeur);

            // Supprimer le caractère à la position du curseur moins un
            newValue.deleteCharAt(cursorPosition - 1);

            // Mettre à jour le texte de l'EditText avec la nouvelle valeur
            ecran_main.setText(newValue.toString());

            // Déplacer le curseur après la suppression
            ecran_main.setSelection(cursorPosition - 1);
        } else {
            // Si la position du curseur est à la première position, ne
rien faire ou effacer tout le texte selon vos besoins
        }
    }
});
```

Methode OnButtonClick : Retourne un String donc c'est-à-dire le caractère du button cliqué. Il contrôle la position de la saisie des nombres et gérer les erreurs.

```
public String onButtonClick(View view) {
    Button button = (Button) view;
    String currentText = ecran_main.getText().toString();
    int cursorPosition = ecran_main.getSelectionStart();

    // Obtenez la partie du texte avant et après la position du curseur
    String textBeforeCursor = currentText.substring(0, cursorPosition);
    String textAfterCursor = currentText.substring(cursorPosition);

    ecran_main.setTextColor(getResources().getColor(android.R.color.black));

    if (ecran_main.getText().equals("Erreur")) {
        ecran_main.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ecran_main.setText("0");
            }
        });
    } else {
        if (currentText.startsWith("0") && (currentText.length() == 1 ||
isOperator(currentText.charAt(1)))) {
            ecran_main.setText(String.format("%s%s%s", textBeforeCursor,
```

```
button.getText(), textAfterCursor));
        } else {
            ecran_main.setText(String.format("%s%s%s", textBeforeCursor,
button.getText(), textAfterCursor));
        }
    }
    // Déplacez le curseur à la fin du texte après l'insertion du chiffre
    ecran_main.setSelection(ecran_main.getText().length());
    ecran_2.setText(calculateResult());

    return ecran_main.getText().toString();
}
```

Le `buton_text` reçoit le caractère de saisie.

Exemple : btn_p1 : “(“

```
btn_p1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        buton_text = onButtonClick(view);
    }
});
```

La methode isOperator permet de verifier si le prochain caractère saisie est un opérateur

```
private boolean isOperator(char character) {
    return character == '+' || character == '-' || character == '*' ||
character == '/';
}
```

La méthode calculateResult ajoute l’opération dans l’écran 2 après chaque operation sans presser le button égale (=)

```
private String calculateResult() {
    try {
        double result = egale(ecran main.getText().toString());
        return String.valueOf(result);
    } catch (Exception e) {
        return "";
    }
}
```

btn_racine : Permet de calculer la racine d’un nombre avec **Math.sqrt**

```
btn_racine.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        try {
            String val = ecran_main.getText().toString();
            double rac = Math.sqrt(Double.parseDouble(val));
            ecran_main.setText(String.valueOf(decimalFormat.format(rac)));
            ecran_2.setText(btn_racine.getText() + "" + val);
        }catch (Exception e){}
    }
});
```

btn_inv : L'inverse du nombre

```
btn_inv.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            ecran_main.setText(ecran_main.getText() + "^" + "(-1)");
        }catch (Exception e){}
    }
});
```

btn_fact : Calcul la factorielle

Quand la valeur saisie est inférieure à zéro on met 0 à l'écran sinon on fait le calcul.

```
btn_fact.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        try {
            double val =
Double.parseDouble(ecran_main.getText().toString());
            facto = factoriel((int) val);
            String r = String.valueOf(facto);
            if (val < 0) {
                ecran_2.setText("0");
                ecran_main.setText("Erreurdomaine");

ecran_main.setTextColor(getResources().getColor(android.R.color.holo_red_da
rk));
            } else if (r.length() <= 11) {
                ecran_main.setText(String.valueOf(facto));
                ecran_2.setText(val + "! ");
            } else ecran_main.setText(val + "! Trop long");
        } catch (Exception e) {
            ecran_2.setText("0");
        }
    }
});
```

Et voici la function qui permet de mieux calculer la factorielle.

```
//function facto
public static BigInteger factoriel(int n) {
    BigInteger result = BigInteger.ONE;
    String r = String.valueOf(result);
    if (r.length() <= 11) {
        for (int i = 1; i <= n; i++) {
            result = result.multiply(BigInteger.valueOf(i));
        }
    }
    return result;
}
```

Elle retourne un nombre de type BigInteger mais on a donné la possibilité de calculer la factorielle d'un nombre dont la longueur de sa factorielle soit inférieur à 11 caractères.

Btn_pow : Calcul le carré d'un nombre. 2^2 = 4

```
btn_pow.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            double d = Double.parseDouble(ecran_main.getText().toString());
            double square = d * d;
            ecran_main.setText(String.valueOf(square));
            ecran_2.setText(d + "²");
        } catch (Exception ex) {  }
    }
});
```

btn_pow_x :

Permet de calculer n'importe quelle puissance, exemple : x^y 2^10 = 1024

```
btn_pow_x.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            String baseText = ecran_main.getText().toString();

            if (!baseText.isEmpty()) {
                baseForPowX = Double.parseDouble(baseText);
                ecran_2.setText(baseForPowX + "^"); // Affiche la base
suivie du signe d'exposant
                ecran_main.setText(""); // Efface l'affichage pour l'entrée
de l'exposant
                waitingForExponent = true; // Indique que l'application
attend l'entrée de l'exposant
            } else {
                // Gérez le cas où la base est vide
                showToast("Veuillez entrer une base avant d'utiliser la
fonction de puissance.");
            }
        } catch (Exception e) { }

    }
});
```

On a ajouté un bouton qui va permettre de mettre la calculatrice en radian ou en grade

```
btn_rd_deg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showToast(btn_rd_deg.getText().toString());
        // Basculez entre radians et degrés
        isRadiansMode = !isRadiansMode;
        // Mettez à jour le texte du bouton en fonction du mode actuel
        btn_rd_deg.setText(isRadiansMode ? "Rad" : "Deg");
    }
});
```

@ La méthode la plus intéressante est l'évènement **btn_equals**

```
btn_egale.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            try {
                if (waitingForExponent) {
                    String exponentText = ecran_main.getText().toString();

                    if (!exponentText.isEmpty()) {
                        double exponent = Double.parseDouble(exponentText);

                        double result = Math.pow(baseForPowX, exponent);

                        ecran_main.setText(String.valueOf(result));
                        ecran_2.setText(baseForPowX + "^" + exponent); //
Efface l'affichage de l'opération de puissance
                        waitingForExponent = false; // Réinitialise le flag
après le calcul
                    } else {
                        showToast("Veuillez entrer un exposant.");
                    }
                }
                String value = ecran_main.getText().toString();
                double resultat = egale(value);
                ecran_main.setText(String.valueOf(resultat));
                ecran_2.setText(value);
            } catch (
                    Exception e) {
                ecran_2.setText("0");
                ecran_main.setText("Erreur");

ecran_main.setTextColor(getResources().getColor(android.R.color.holo_red_da
rk));
                showToast("Erreur de Calcul");
            }
        }
    });
} catch (
        Exception ex) {
    ecran_main.setText("");
}
```

- ⇨ Cette partie vérifie la partie puissance (base et exposant)

```
if (waitingForExponent) {
      String exponentText = ecran_main.getText().toString();

         if (!exponentText.isEmpty()) {
             double exponent = Double.parseDouble(exponentText);
             double result = Math.pow(baseForPowX, exponent);

             ecran_main.setText(String.valueOf(result));
             ecran_2.setText(baseForPowX + "^" + exponent);
            // Efface l'affichage de l'opération de puissance
             waitingForExponent = false;
            // Réinitialise le flag après le calcul
         } else {
```

```
                showToast("Veuillez entrer un exposant.");
            }
        }
```

La fonction qui fait les calculs :

```
public static double egale(final String str) {
    return new Object() {
        int pos = -1, ch;
        void nextChar() {
            try {
                ch = (++pos < str.length()) ? str.charAt(pos) : -1;
            } catch (Exception e) {
                Toast.makeText(ecran_main.getContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        }

        boolean eat(int charToEat) {
            try {
                while (ch == ' ') nextChar();
                if (ch == charToEat) {
                    nextChar();
                    return true;
                }
            } catch (Exception e) {
                Toast.makeText(ecran_main.getContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
            return false;
        }

        double parse() {
            nextChar();
            double x = parseExpression();
            try {
                if (pos < str.length()) {
                    throw new RuntimeException();
                }
            } catch (Exception e) {
                ecran_main.setText("");
            }
            return x;
        }

        double parseExpression() {
            double x = parseTerm();
            for (; ; ) {
                if (eat('+')) {
                    if (isOperatorNext()) {
                        throw new RuntimeException("Unexpected: " + (char)
ch);
                    }
                    x += parseTerm(); // addition
                } else if (eat('-')) {
                    if (isOperatorNext()) {
                        throw new RuntimeException("Unexpected: " + (char)
ch);
                    }
                    x -= parseTerm(); // subtraction
```

```
                } else {
                    return x;
                }
            }
        }

        boolean isOperatorNext() {
            int nextPos = pos;
            while (nextPos < str.length() && str.charAt(nextPos) == ' ') {
                nextPos++;
            }
            return (nextPos < str.length()) &&
                    (str.charAt(nextPos) == '+' || str.charAt(nextPos) ==
'-'
                            || str.charAt(nextPos) == '*' ||
str.charAt(nextPos) == '/');
        }

        double parseTerm() {
            double x = parseFactor();
            for (; ; ) {
                if (eat('*')) {
                    x *= parseFactor(); // multiplication
                } else if (eat('/')) {
                    x /= parseFactor(); // division
                } else if (ch == '(') {
                    // Ajout de la multiplication implicite avec les
parenthèses
                    x *= parseFactor();
                } else if (ch >= 'a' && ch <= 'z') {
                    x *= parseFactor(); // Prendre en compte le facteur
actuel
                    x = parseFactor(); // Parsez la fonction
trigonométrique
                } else {
                    return x;
                }
            }
        }

        double parseFactor() {
            if (eat('+')) return parseFactor(); // unary plus
            if (eat('-')) return -parseFactor(); // unary minus

            double x;
            int startPos = this.pos;
            if (eat('(')) { // parentheses
                x = parseExpression();
                eat(')');
            } else if ((ch >= '0' && ch <= '9') || ch == '.') { // numbers
                while ((ch >= '0' && ch <= '9') || ch == '.') nextChar();
                x = Double.parseDouble(str.substring(startPos, this.pos));
            } else if (ch >= 'a' && ch <= 'z') { // functions
                while (ch >= 'a' && ch <= 'z') nextChar();
                String func = str.substring(startPos, this.pos);
                x = parseFactor();
                if (func.equals("sqrt")) x = Math.sqrt(x);
                else if (func.equals("sin")) {
                    x = isRadiansMode ? Math.sin(Math.toRadians(x)) :
Math.sin(x);
                } else if (func.equals("cos")) {
```

```
                    x = isRadiansMode ? Math.cos(Math.toRadians(x)) :
Math.cos(x);
                } else if (func.equals("tan")) {
                    x = isRadiansMode ? Math.tan(Math.toRadians(x)) :
Math.tan(x);
                } else if (func.equals("log")) x = Math.log10(x);
                else if (func.equals("ln")) x = Math.log(x);
                else {
                    throw new RuntimeException("Unknown function: " + func);
                }
            } else {
                throw new RuntimeException("Unexpected: " + (char) ch);
            }
            // Nouvelle vérification pour les constantes multiplicatives
            while (Character.isDigit(ch)) {
                // Consommer les chiffres pour former une constante
                nextChar();
            }

            if (eat('*')) {
                // S'il y a un '*' après la constante, multipliez par le
résultat de parseFactor()
                x *= parseFactor();
            } else if (eat('^')) {
                // Exponentiation après la constante
                x = Math.pow(x, parseFactor());
            }
            return x;
        }
    }.parse();
}
```

- ⇨ Elle gère tous les calculs qui concernent les sinus, cosinus, tangente, ln, logarithme, c’est la base même du projet.

On a créé cette methode **showToast** afin d’afficher le Toast semblable.

```
private void showToast(String message) {
    SpannableString spannableString = new SpannableString(message);
    //spannableString.setSpan(new
BackgroundColorSpan(Color.parseColor("#57CDCA"), 0, message.length(), 0);
    spannableString.setSpan(new ForegroundColorSpan(Color.WHITE), 0,
message.length(), 0);

    // Créez le Toast personnalisé
    Toast toast = Toast.makeText(this, spannableString, Toast.LENGTH_SHORT);
    toast.show();
}
```

Cycle de vie du projet

```
//Cycle de vie
@Override
protected void onStop() {
    super.onStop();
    showToast("Calculator on Stop");
}

@Override
protected void onResume() {
    super.onResume();
    showToast("Calculator on Resume");
}
@Override
protected void onPause() {
    super.onPause();
    showToast("Calculator on Pause");
}
@Override
protected void onDestroy() {
    super.onDestroy();
    showToast("Calculatrice on Destroy");
}

@Override
protected void onStart() {
    super.onStart();
    showToast("Calculatrice on Start");
}
```

Remerciement

Cher Professeur Foko,

Nous tenons à exprimer notre sincère gratitude pour cette session passée avec vous en tant qu'étudiants. Votre dévouement, vos encouragements et votre expertise ont été essentiels à notre formation académique. Grâce à votre passion communicative, nous avons acquis non seulement des connaissances approfondies dans votre matière, mais aussi une inspiration durable pour la poursuite de l'apprentissage. Merci pour avoir été bien plus qu'un enseignant, mais aussi un guide et une source d'inspiration. Cette année restera gravée dans notre mémoire, en grande partie grâce à votre influence positive.

Cordialement,

INUKA'S STUDENTS