OOP'S PROJECT...

SUBMITTED BY

01

DIVYAM 21103046

02

FARHAN AKHTAR 21103047

03

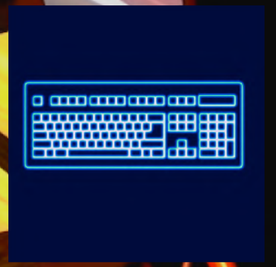
GURSHAANPREET SINGH 21103054

SUBMITTED TO

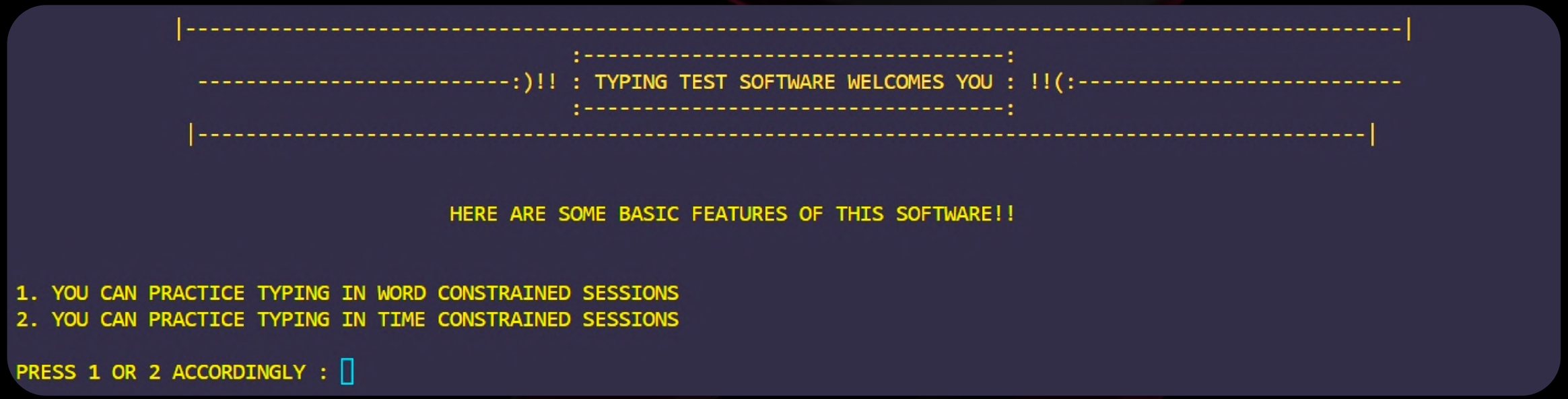
ER. RAHUL AGGARWAL

TYPING TEST

SOFTWARE...



INTERFACE



01

"WORD CONSTRAINT

→Test your typing speed in no limited time.

02

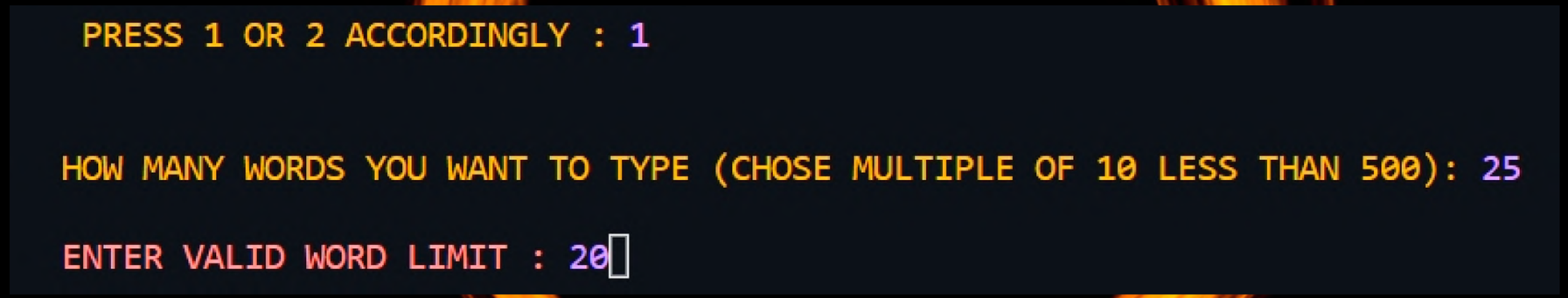
"TIME CONSTRAINT

→Test your typing speed in limited time limit.

PROJECT INSIGHT_

WORD CONSTRAINT SESSION

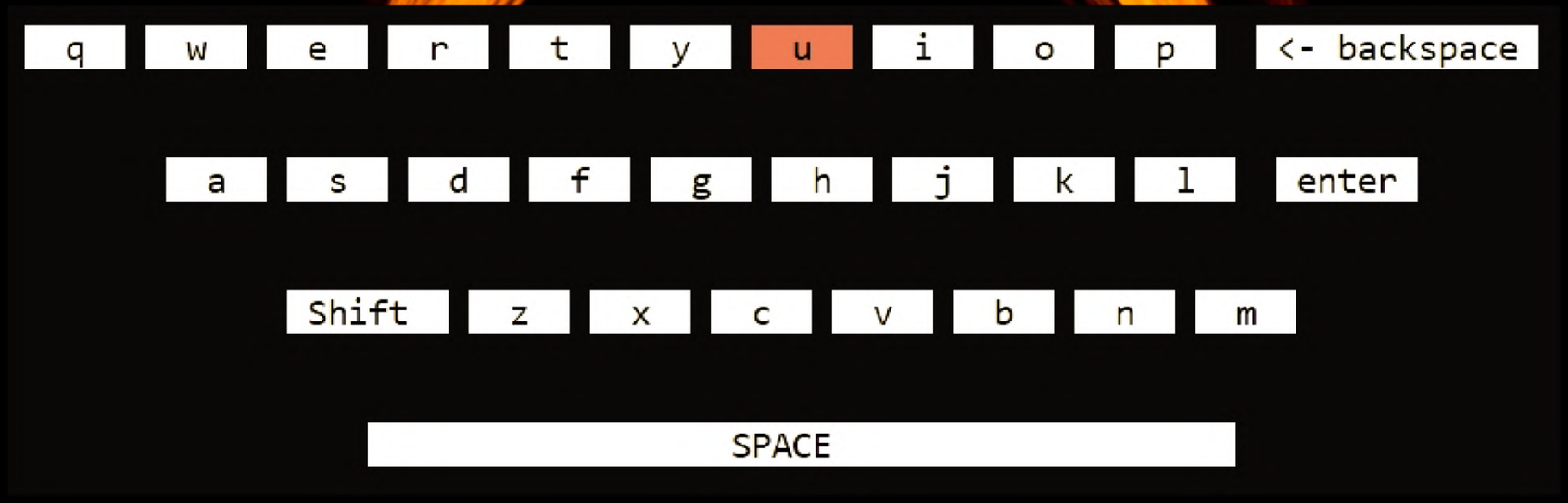
I)



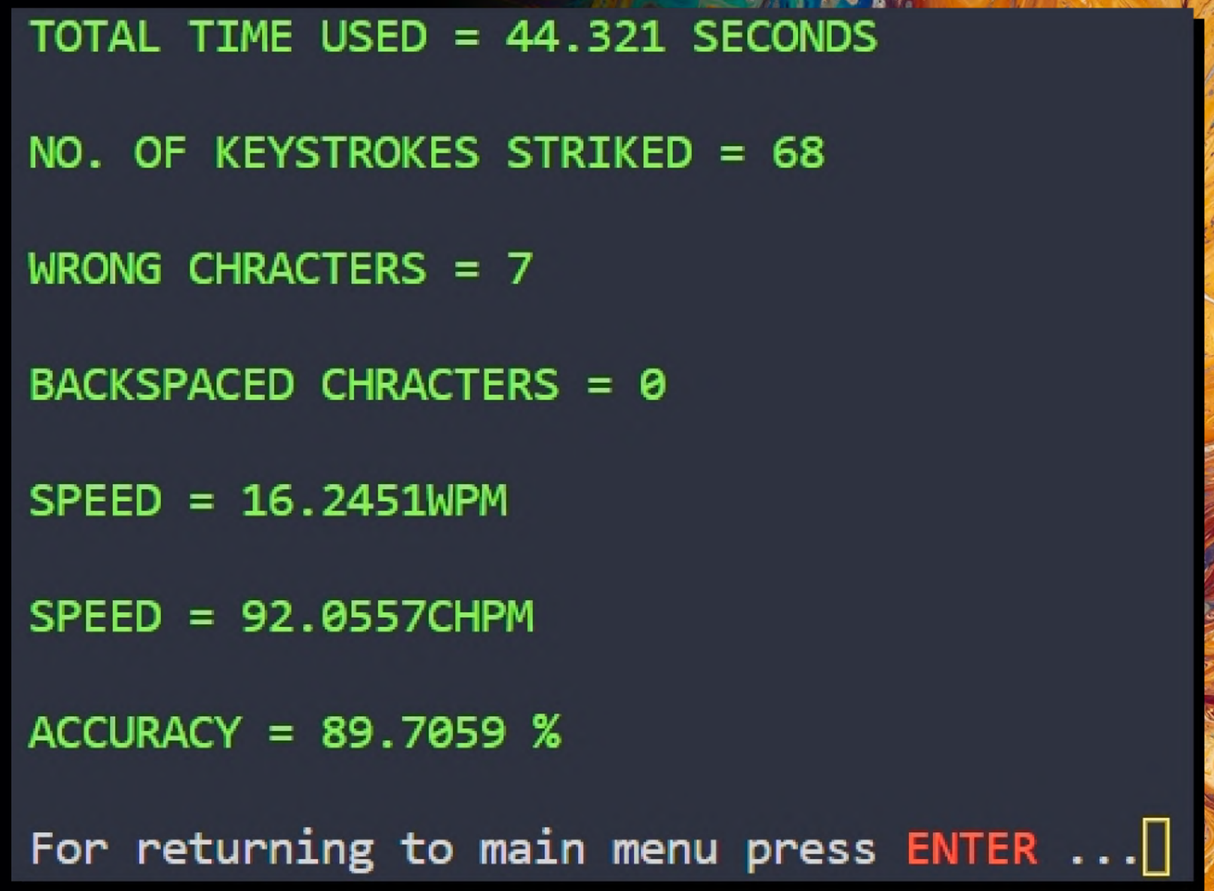
II)

even posture cunning cheek electronics radical sandwich available nightmare movement

aloof blonde huge feature feminine secretary salvation fork aloof trolley



REPORT :


TOTAL TIME USED = 44.321 SECONDS
NO. OF KEYSTROKES STRIKED = 68
WRONG CHRACTERS = 7
BACKSPACED CHRACTERS = 0
SPEED = 16.2451WPM
SPEED = 92.0557CHPM
ACCURACY = 89.7059 %
For returning to main menu press ENTER ...

TIME CONSTRAINT SESSION

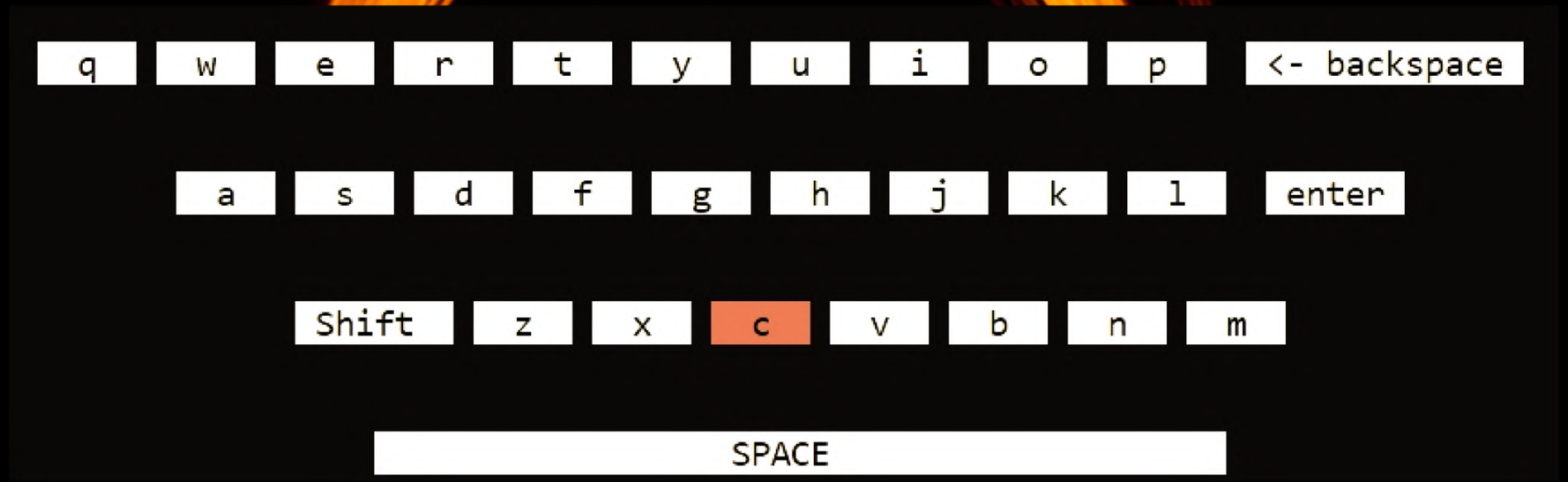
I)

```
PRESS 1 OR 2 ACCORDINGLY : 2

                    THERE ARE 3 MODES IN TIME CONSTRAINED SESSION :
                (YOU CAN WRITE ANY NUMBER OF WORDS IN THE GIVEN TIME)

                                    1. 30 SEC TIMER
                                    2. 1 MIN TIMER
                                    3. 2 MIN TIMER


                              PRESS MODE NUMBER ACCORDINGLY : 4

                                    ENTER VALID MODE : 2
```

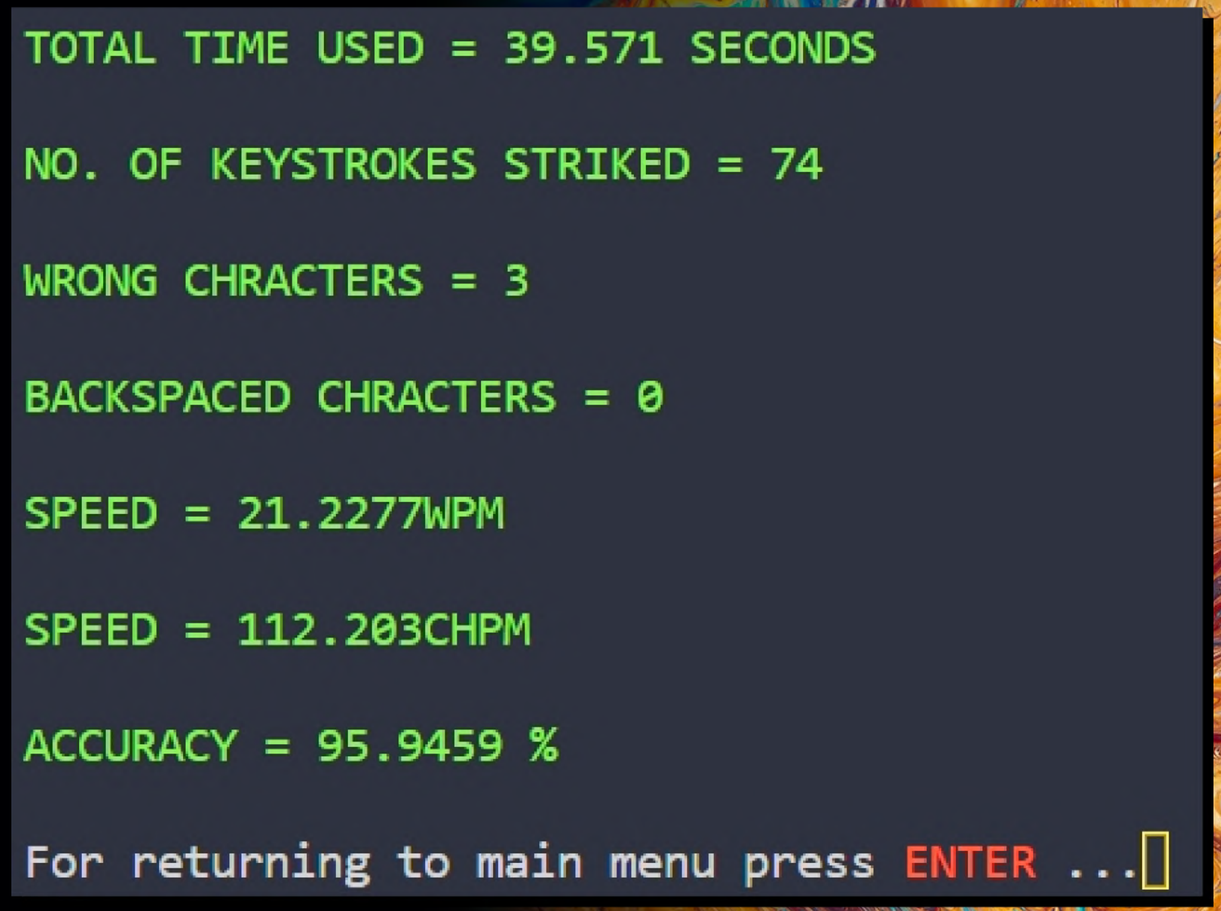
II)

celebrate autonomy balloons goal feninene clearance really purduit knowledge slim

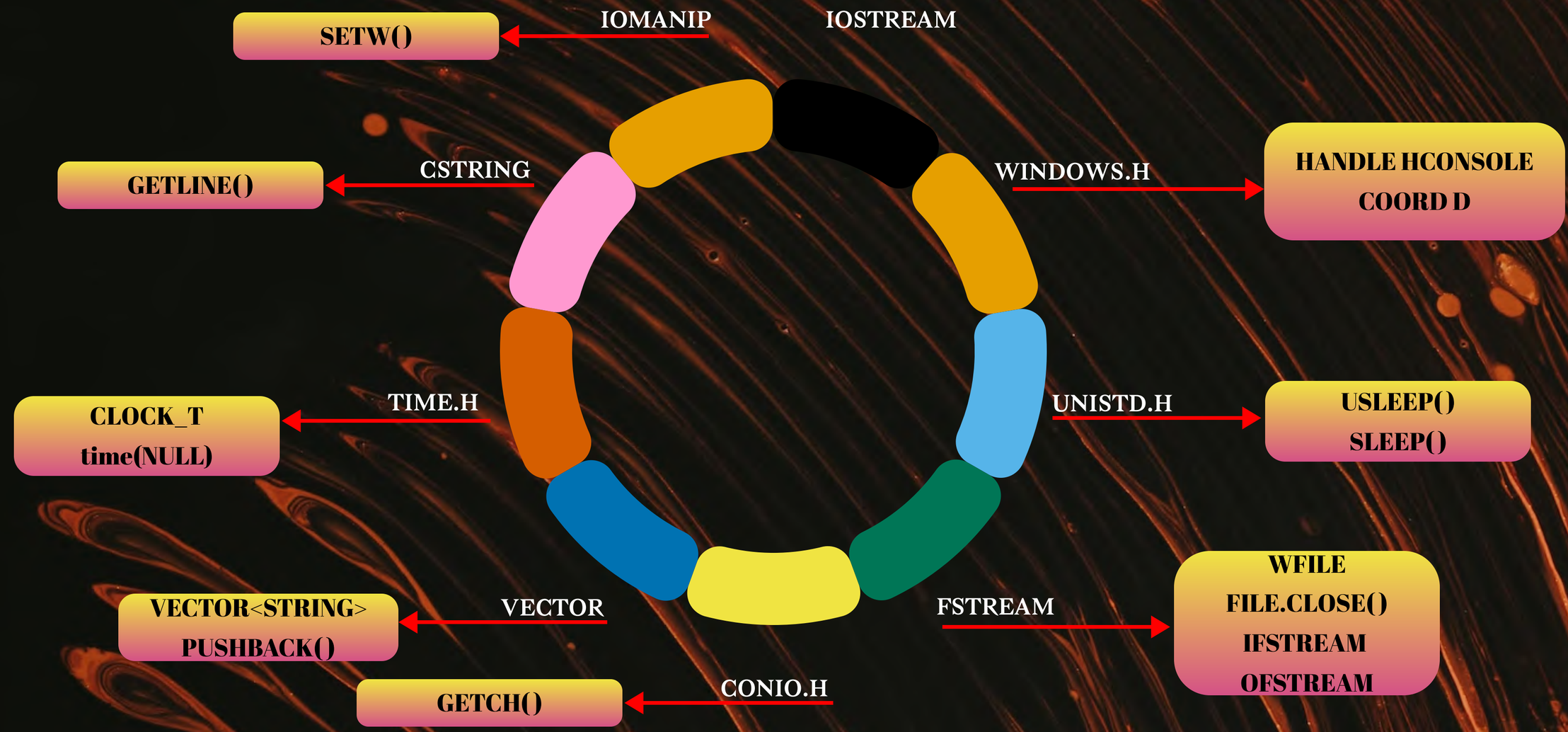
regard expertise elephant mistake communist tap marriage chai



REPORT:


TOTAL TIME USED = 39.571 SECONDS
NO. OF KEYSTROKES STRIKED = 74
WRONG CHRACTERS = 3
BACKSPACED CHRACTERS = 0
SPEED = 21.2277WPM
SPEED = 112.203CHPM
ACCURACY = 95.9459 %
For returning to main menu press ENTER ...

HEADER FILES :



OOP'S TERMINOLOGIES/.\

INHERITANCE

```
class Sentence_formation : public Typing_Test
```

FUNCTION OVERLOADING

```
void check(char c[], char input[], int j);
void check(char c[], int times);
```

STL

```
vector<string> lines;
lines.push_back(line);
```

MANIPULATORS

```
cout << setw(93)
cout << setw(70)
cout << setw(69)
cout << setw(71)
cout << setw(75)
```

FILE HANDLING

```
ofstream wfile("write.txt", ios::app);
```

SYNOPSIS**

First of all, we made a main base class that is Keyboard_single_line

```
class keyboard_single_line
{
public:
    void keyboard_single_line::set(int c, int j, int k);
    void keyboard_single_line::print_ist_part(string s, string f, int length);
    void keyboard_single_line::print_2nd_part(int i, string s, int length);
    void keyboard_single_line::printkeys(char arr[], int i);
    void keyboard_single_line::print_main_keyboard(char arr[]);
    void keyboard_single_line::check_key_pressed(char arr[], char c, int i);
    void keyboard_single_line::print_red_key(char arr[], char c);
};
```

BASE CLASS

Keyboard_single_line : this class is responsible for displaying virtual 🎹 keyboard..

```
class Typing_Test : protected keyboard_single_line
{
protected:
    clock_t start, end;
    double time_used;
    void set_parameters();
    void check(char c[], char input[], int j);
    void check(char c[], int times);
    void check_for_time(char c[]);

public:
    string fin_inp = "";
    int length = 0;
    int count = 0;
    int checkcount = 0;
    int wrong_chracters = 0;
    int count_back = 0;
};
```

INHERITANCE CHILD CLASS

Typing_Test : this class is made that will gives basically result of the user input, check for the error in both word and time constraint and adding colors to the user input...

Secondally, we made two class i.e Typing_Test and Sentence_formation inherited from base class

Sentence_formation : this class is responsible for making time limit constraint, displaying 10 word per instant and handling to store history of the user result...

```
class Sentence_formation : public Typing_Test
{
    string line;
    vector<string> lines;

public:
    int process_count = 0;
    int session_count = 0;
    int round_count = 0;
    void store_words(int a, float tym);
    void process(int c, int d);
    void word_const();
    void time_const();
};
```

INHERITANCE GRAND CHILD CLASS

KEYBOARD_SINGLE_LINE:

```
void set(int c, int j, int k)
{

    HANDLE hconsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hconsole, c);
    COORD d;
    d.X = 20 + j;
    d.Y = 12 + k;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), d);
}
```

Assigning the co-ordinates and setting the colors of the output terminal.

This func is responsible for giving the colors to the user input.

```
void print_ist_part(string s, string f, int length)
{
    for (int j = 0; j < length; j++)
    {
        if (f[j] == s[j])
        {

            set(2, -5 + j, -10);
            cout << s[j];
            cout << "\xDB";
        }
        else
        {
            set(4, -5 + j, -10);
            cout << s[j];
            cout << "\xDB";
        }
    }
}
```

```
void print_2nd_part(int i, string s, int length)
{
    for (int j = i + 1; j < length; j++)
    {
        set(0, -5 + j, -10);
        cout << s[j];
    }
}
```

Making the untyped string remains GREY.

This func is printing the virtual keyboard and it's keys on screen.

```
void printkeys(char arr[], int i)
{
    HANDLE hconsole = GetStdHandle(STD_OUTPUT_HANDL
    SetConsoleTextAttribute(hconsole, 134);
    cout << "   " << arr[i] << "   ";
    SetConsoleTextAttribute(hconsole, 1);
    cout << " ";
}
```

```
void print_main_keyboard(char arr[])
{
    HANDLE hconsole = GetStdHandle(STD_OUTPUT_HANDLE);
    printkeys(arr, 16);
    printkeys(arr, 22);
    printkeys(arr, 4);
    printkeys(arr, 17);
    printkeys(arr, 19);
    printkeys(arr, 24);
    printkeys(arr, 20);
    printkeys(arr, 8);
    printkeys(arr, 14);
    printkeys(arr, 15);
    cout << " ";
    SetConsoleTextAttribute(hconsole, 134);
    cout << " <- backspace " << arr[28] << " ";
    SetConsoleTextAttribute(hconsole, 1);
    cout << "    \n\n\n                                    ";
    printkeys(arr, 0);
    printkeys(arr, 18);
```

```
printkeys(arr, 10);
printkeys(arr, 11);
cout << " ";
SetConsoleTextAttribute(hconsole, 134);
cout << " enter ";
SetConsoleTextAttribute(hconsole, 1);
cout << "    \n\n\n                                ";
cout << "       ";
SetConsoleTextAttribute(hconsole, 134);
cout << " Shift  ";
SetConsoleTextAttribute(hconsole, 1);
cout << " ";
printkeys(arr, 25);
printkeys(arr, 23);
printkeys(arr, 2);
printkeys(arr, 21);
printkeys(arr, 1);
printkeys(arr, 13);
printkeys(arr, 12);
cout << " \n\n\n                                        ";
SetConsoleTextAttribute(hconsole, 134);
cout << "                    SPACE                    ";
SetConsoleTextAttribute(hconsole, 1);
cout << "    \n\n\n                                ";
```

This func is responsible for displaying our virtual keyboard ..

```
void check_key_pressed(char arr[], char c, int i)
{
    HANDLE hconsole = GetStdHandle(STD_OUTPUT_HANDLE)

    if (c - 97 == i)
    {
        SetConsoleTextAttribute(hconsole, 192);
        cout << "  " << arr[i] << "  ";
        SetConsoleTextAttribute(hconsole, 1);
        cout << " ";
    }
    else
    {

        SetConsoleTextAttribute(hconsole, 134);
        cout << "  " << arr[i] << "  ";
        SetConsoleTextAttribute(hconsole, 1);
        cout << " ";
    }
}
```

```
void print_red_key(char arr[], char c)
{
    HANDLE hconsole = GetStdHandle(STD_OUTPUT_HANDLE);
```

These both functions help in analyzing the real keyboard input with displayed keyboard and giving it the RED color while tapping while other remains WHITE.

TYPING_TEST :

```
void Typing_Test::check(char c[], char input[], int j)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    set_parameters();
    cout << c << endl;
    for (int i = 0; i <= j; i++)
    {
        if (c[i] == input[i])
        {
            SetConsoleTextAttribute(hConsole, 2);
            cout << input[i];
        }
        else
        {
            SetConsoleTextAttribute(hConsole, 4);
            cout << input[i];
        }
    }
}
```

This func is made to add colors to the user input in two ways:

1) making GREEN letters iff user input matches with the given displayed string.

2) making RED letters iff user input doesn't matches the given displayed string.

FUNCTION OVERLOADING

Here three parameter check func will check for the color of the input if backspace is done.

```
void Typing_Test ::check(char c[], int times)
{
    checkcount++;
    if (checkcount == 1)
    {
        start = clock();
    }
    char input[strlen(c)];
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    for (int i = 0; i < strlen(c) - 1; i++)
    {
    label:
        input[i] = getch();
        if (input[i] == '\r')
        {
            length--;
            break;
        }
        if (input[i] == 8)
        {

            length--;
            set_parameters();
            cout << c << endl;
            check(c, input, i);
            i--;
            count_back++;
            goto label;
        }
```

This check( ) func is made to calculate wrong characters, calculating the time taken by the user to input in WORD constraint session..

Time is calculated by taking out the difference b/w the starting point and ending point of the user input.

```
        else
        {
            length++;
            if (input[i] == c[i])
            {
                count++;
                SetConsoleTextAttribute(hConsole, 2);
                cout << input[i];
            }
            else
            {
                SetConsoleTextAttribute(hConsole, 4);
                cout << input[i];
            }
        }
    }
    if (checkcount == times)
    {
        end = clock();
    }
    if (length == strlen(c) - 1)
    {
        wrong_chracters = (length + count_back) - count;
    }
    else
    {
        wrong_chracters = (length + 1 + count_back) - count;
    }
}
```

```
void Typing_Test::check_for_time(char c[])
{
    char input[strlen(c)];
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    for (int i = 0; i < strlen(c) - 1; i++)
    {
    label:
        if (time(NULL) > end_time)
        {
            length--;
            break;
        }
        input[i] = getch();
        if (input[i] == '\r')
        {
            break;
        }
        if (time(NULL) > end_time)
        {
            break;
        }
        if (input[i] == 8)
        {
            length--;
            set_parameters();
            cout << c << endl;
            check(c, input, i);
            i--;
            count_back++;
            goto label;
```

This check_for_time( ) func is made to calculate wrong characters, synchronizing the time limit for the user to input in TIME constraint session..

Time Constraint is generated by taking out the difference b/w the two instance and using while loop condition for stopping the limited time.

SENTENCE_FORMATION :

Time_const( ) func is made to start the starting of TIME constraint session and the user input is tailing according to this func only after that it will over the time limit automatically.

```
void Sentence_formation::time_const()
{
    while (time(NULL) < end_time)
    {
        process(2, 0);
    }
    cout << "\n  TIMES OVER ";
}
```

```
void Sentence_formation::word_const()
{
    int t;
    cout << "\x1B[33m"
         << "         \n\n   HOW MANY WORDS YOU WANT TO TYPE
         (CHOOSE MULTIPLE OF 10 LESS THAN 500) : "
         << "\x1B[0m";
choice:
    cout << "\x1B[35m";
    cin >> t;

    if (t % 10 != 0 || t == 0 || t > 500)
    {
        cout << "\x1B[31m"
             << "\n   ENTER VALID WORD LIMIT : "
             << "\x1B[0m";
        goto choice;
    }
    int b = t / 10;

    while (b--)
    {
        process(1, t / 10);
    }
}
```

Word_const( ) func is made to initialize the starting of WORD constraint session and sending the order of no. of 10 word sentences to be formed to the process( ) func.

Process( ) func is responsible for counting the session of the user and generating 10 words per sentences using random function and displaying one by one before user after user attempt.

```
void Sentence_formation::process(int c, int d)
{
    process_count++;
    system("cls");
    string s;

    srand(time(0));
    for (int i = 0; i < 10; i++)
    {

        int random_number = rand() % 500;
        s += lines[random_number] + " ";
    }

    char arr[s.length()];

    for (int i = 0; i <= s.length(); i++)
    {
        arr[i] = s[i];
    }

    set_parameters();
```

Also in Process( ) func it is displaying the starting timer for the user to be ready for the test and sending the user input to check( ) and check_for_time( ) func for generating the report.

INHERITED FUNCTION

```
if (process_count == 1)
{
    for (int i = 3; i > 0; i--)
    {
        cout << s;
        cout << "\n\nYour Typing Test will start in "
             << "\x1B[31m" << i << "\x1B[0m"
             << " seconds...";
        sleep(1);
        system("cls");
    }
}
cout << s;
cout << endl;
if (c == 1)
{

    check(arr, d);
    cout << endl;
}
else
{
    check_for_time(arr);
    cout << endl;
}
```

```
void Sentence_formation ::store_words(int a, float tym)
{
    session_count++;
    round_count++;
    ifstream file("p.txt");

    while (getline(file, line))
    {

        lines.push_back(line);
    }
    if (a == 1)
    {
        word_const();
        set_parameters();
        ofstream wfile("write.txt", ios::app);
        time_used = (((double)(end - start)) / CLOCKS_PER_SEC);
        wfile << "\n\n\n\nSESSION " << session_count;
        wfile << "\n\nROUND " << round_count;
        wfile << "\n\nTOTAL TIME USED = " << time_used << " SECONDS";
        wfile << "\n\nNO. OF KEYSTROKES STRIKED = " << length + 1;
        wfile << "\n\nWRONG CHRACTERS = " << wrong_chracters;
        wfile << "\n\nBACKSPACED CHRACTERS = " << count_back;
        wfile << "\n\nSPEED = " << (((length + 1) / 10) / (time_used / 60)) * 2 << "WPM ";
        wfile << "\n\nSPEED = " << (((length + 1)) / (time_used / 60)) << "CHPM ";
        wfile << "\n\nACCURACY = " << (double)((double)((double)(length + 1 - count_back - wrong_chracters) /
        (double)(length + 1))) * 100 << " % ";
```

Store_words( ) is responsible for storing all the report of the user in the FILE and also displaying the report too.

FILE HANDLING

EXTRA FUNC ( );

```
void clear_()
{
    ofstream file("write.txt", ios::out);
    file << "
         << "
         << "
}
```

```
 ----------------------------------------\n"
|::::::::::HISTORY OF LAST SESSIONS:::::::|\n"
 ----------------------------------------\n";
```

Through this we are storing the history of last sessions and all the previous session will be automatically cleared..

write.txt

```
1
2
3
4
```

```
 ----------------------------------------
|::::::::::HISTORY OF LAST SESSIONS:::::::|
 ----------------------------------------
```

EXTRA FUNC ( );

```
int set_keys()
{
    for (int i = 0; i < 26; i++)
    {
        arr[i] = 97 + i;
    }
    arr[26] = 10;
    arr[27] = 32;
    arr[28] = 8;
    arr[29] = 13;
}
```

This func is globally declared and responsible for storing the virtual keyboard character in the array according to their ASCII code .


..THANK YOU..