

## A Human Exercise Machine Feed-back Program to Predict the Fatigue Point of An Untrained Person

The program was developed to integrate into a human exercise machine. It can predict the fatigue point of an untrained person before approximately 30 seconds while the person conducts the exercise on the machine. The program uses heart rate variability and heart rate to predict the fatigue point. The code is written in the C programming language. The code and screenshots are provided below.

```
#include <stdio.h>

#include <math.h>


int RRi=0, RRi_1=0;    //two consecutive RR time intervals
int rri=0, rri_1=0;
float RMSSD1=0;        //variable to store RMSSD value
float RMSSD2=0;
float RMSSDG=0;        //variable to store RMSSD gradient
float HRG=0;           //heart rate gradient value
float a=0.03;          //a constant value that defines a test value for HRV gradient before a
person get fatigued
float b=-0.2;          //a constant value that defines a test value for HR gradient before a
person get fatigued
int elapsed_time=0;    //time elapsed for selected RR time intervals
int m=40;              //Maximum user input RR time interval amount
int N=0;               //Number of RR intervals (this is an input from the user)
int sq_of_difference=0; //squared of RR interval difference
float sum_of_sq_difference=0; //sum of squared of consecutive RR time interval differences
float sq_mean_sum=0;   // mean of sum of squared RR time interval differences
int sq_difference=0;   //squared of RR interval difference
float sum_sq_difference=0; //sum of squared of consecutive RR time interval differences
float mean_sum=0;      // mean of sum of squared RR time interval differences
```

```
int pulse_data[40]={715,720,730,740,750,760,770,780,790,800,810,820,830,840,850,860,870,
880,890,900,910,920,930,940,950,960,970,980,990,1000,1010,1020,1030,1040,1050,1060,107
0,1080,1090,1100};
```

```
// array to store consecutive RR time interval values (in milliseconds) between pulses (user can
modify this array in the main function when the code is used in real-time
```

```
int defining_array(){
    printf("Enter number of RR intervals = ");
    //user enter the number of RR time intervals needed to measure RMSSD
    scanf("%d",&N);
    if (N>m/2){
        printf("%s %d %s","Please enter a value less than ",m/2,"!");
    }
    return N;
}
```

```
float RMSSD_calc(){
    N=defining_array();
    if (N<m/2){
        int time_interval_array[2*N];
        //array that stores RR time interval values
        for(int j=0;j<2*N;j++){
            time_interval_array [j]=pulse_data[j];
            //store RR time intervals in an array
        }
        for (int i=0;i<N-1;i++){
            R Ri=time_interval_array[i];
```

```

    R Ri_1=time_interval_array[i+1];
    sq_of_difference= (R Ri-R Ri_1)*(R Ri-R Ri_1);
    //squared of the difference between i-th and i+1-th RR time interval
    sum_of_sq_difference=sum_of_sq_difference+sq_of_difference;
    //taking sum of squared of the RR time interval difference
    sq_mean_sum=sum_of_sq_difference/(N-1);    //mean of the sum
    RMSSD1=sqrt(sq_mean_sum);                  //Square root of the mean
}
for (int p=N;p<2*N-1;p++){
    rri=time_interval_array[p];
    rri_1=time_interval_array[p+1];
    sq_difference= (rri-rri_1)*(rri-rri_1);
    //squared of the difference between i-th and i+1-th RR time interval
    sum_sq_difference=sum_sq_difference+sq_difference;
    //taking sum of squared of the RR time interval difference
    mean_sum=sum_sq_difference/(N-1);          //mean of the sum
    RMSSD2=sqrt(mean_sum);                     //Square root of the mean
}
for (int q=0; q<N;q++){
    elapsed_time= elapsed_time+time_interval_array[q];    //calculating the elapsed time
}
RMSSDG= (RMSSD1-RMSSD2)/elapsed_time;          //RMSSD gradient
return RMSSDG;
}
}

```

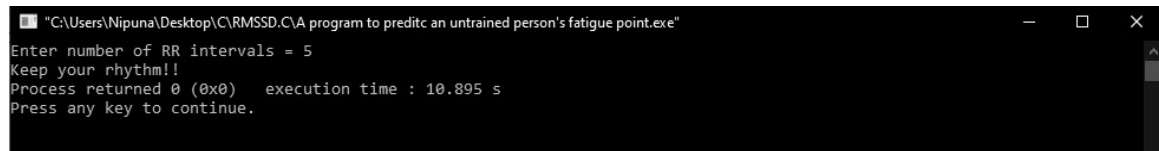
```

float HR(){
    if (N<m/2){
        int HR1[N];                //arrays to store time interval data
        int HR2[N];
        float hr1=0;                //variable to store elapsed time
        float hr2=0;
        for (int k=0; k<N;k++){
            HR1[k]=pulse_data[k];
            HR2[k]=pulse_data[k+N];
            hr1=hr1+HR1[k];          //calculating elapsed time
            hr2=hr2+HR2[k];
        }
        HRG=((1/hr1)-(1/hr2))*N*60000;    //calculated heart rate gradient
        return HRG;
    }
}

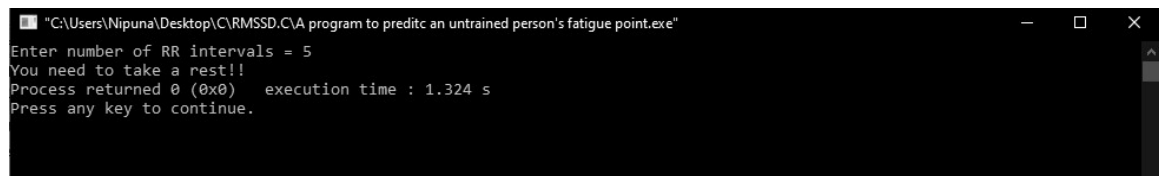
int main(){
    RMSSD_calc();
    HR();
    if ((HRG<b)&&(RMSSDG>a)){
        printf("You need to take a rest!!");
        //condition that predicts human fatigue prior to approximately 30 seconds
    }
    else{
        printf("Keep your rhythm!!");
    }
}

```

Output screenshots of the program at two different heart rate variability and heart rate values



```
"C:\Users\Nipuna\Desktop\C\RMSSD.C\A program to predict an untrained person's fatigue point.exe"
Enter number of RR intervals = 5
Keep your rhythm!!
Process returned 0 (0x0)   execution time : 10.895 s
Press any key to continue.
```



```
"C:\Users\Nipuna\Desktop\C\RMSSD.C\A program to predict an untrained person's fatigue point.exe"
Enter number of RR intervals = 5
You need to take a rest!!
Process returned 0 (0x0)   execution time : 1.324 s
Press any key to continue.
```