BIEN 3200: Lab 3

Decision Statements, Pointers, DMA, and Event Detection Algorithms

Peter Dobbs
Lab Section 401
Nicholas Heugel

4 October 2016

Abstract:

It is of great importance to understand the role of signal processing in the field of electrophysiology. As students of biomedical engineering, this class needs to experience this relationship between systems and signals and how to better understand and model the systems and signals. In order to study the electrophysiological signal of an ECG for the purposes of the QRS complex, an event detection algorithm must be developed. The algorithm can then be customized for a patient, based on the current state of the patient. With the results of the detection algorithm, the average heart rate can then be determined.

Introduction:

This lab was conducted to provide more practice with coding standards, to introduce dynamic memory allocation (DMA), and to test an event detection algorithm in the context of signal processing for an electrocardiogram. The event detection algorithm was optimized to detect QRS complexes for the provided sample data. With the knowledge of the QRS complex, the average heart rate of the sample was determined.

Methods:

Data Files:

Sinus4.txt (1)

The single data file for this lab contains sinus rhythm (normal) ECG data sampled at 250 Hz for 10 seconds. The data is organized in two columns, one being time in units of seconds and the second being amplitude in units of millivolts.

In order to find the number of QRS complexes, an event detection algorithm must be implemented. A threshold amplitude, based on a user defined percentage of the maximum ECG amplitude (ECG_{max}), is required for defining the event of a QRS complex. A blanking period,

also from user input, is used for speeding up the algorithm by skipping over irrelevant data. The threshold crossing locations (in seconds) and the amplitudes (in millivolts) are saved for later use in a user specified output file. Then, based on the number of threshold crossings and the length of the ECG recording (10 seconds, in the case of the provided data), the average heart rate (in beats per minute) can be determined.

For optimization of this algorithm, two simple methods can be employed. First, to maximize the number of correctly detected QRS complexes, keep the blanking period constant and vary the amplitude threshold starting around 50% and increasing the threshold until the optimal percentage is found. The optimal threshold can be confirmed by varying the blanking period between 0.1 and 0.3 seconds and making sure the number of threshold crossings is still the correct amount.

Results:

```
Enter input file name
sinus4.txt
number of data points in file = 2500
ECGmax = 434.320007
Enter a threshold percentage of ECGmax
.7
Enter a blanking period between 0.1 and 0.3 seconds
.3
Enter ouput file name
out.txt
Average Heart Rate (beats/minute) = 90.036011
Press any key to continue . . .
```

Figure 1: Screenshot of the output for the EventDetection.cpp program. The output shows that the average heart rate for the input data set is 90.036011 beats per minute.

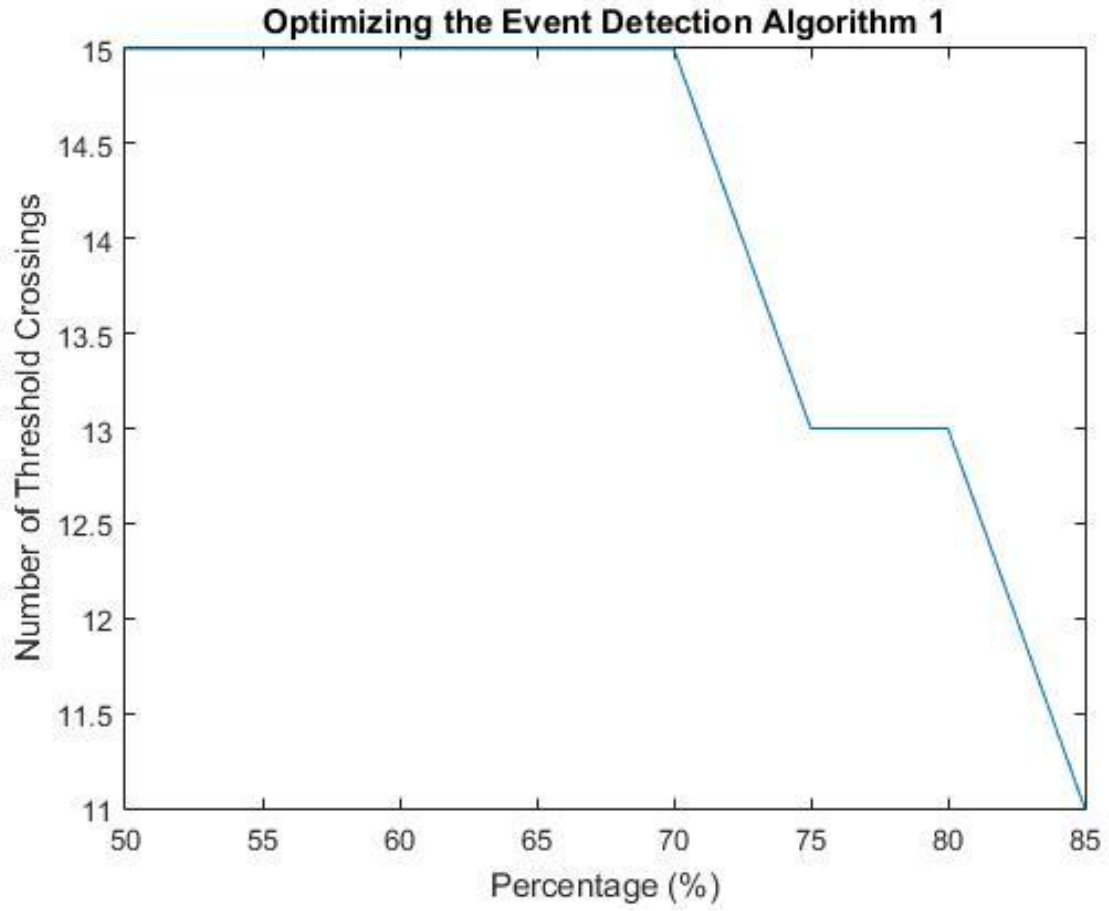


Figure 2: Plot of resulting number of crossings for given percentages of the ECG_{max} with a blanking period of 0.3 seconds.

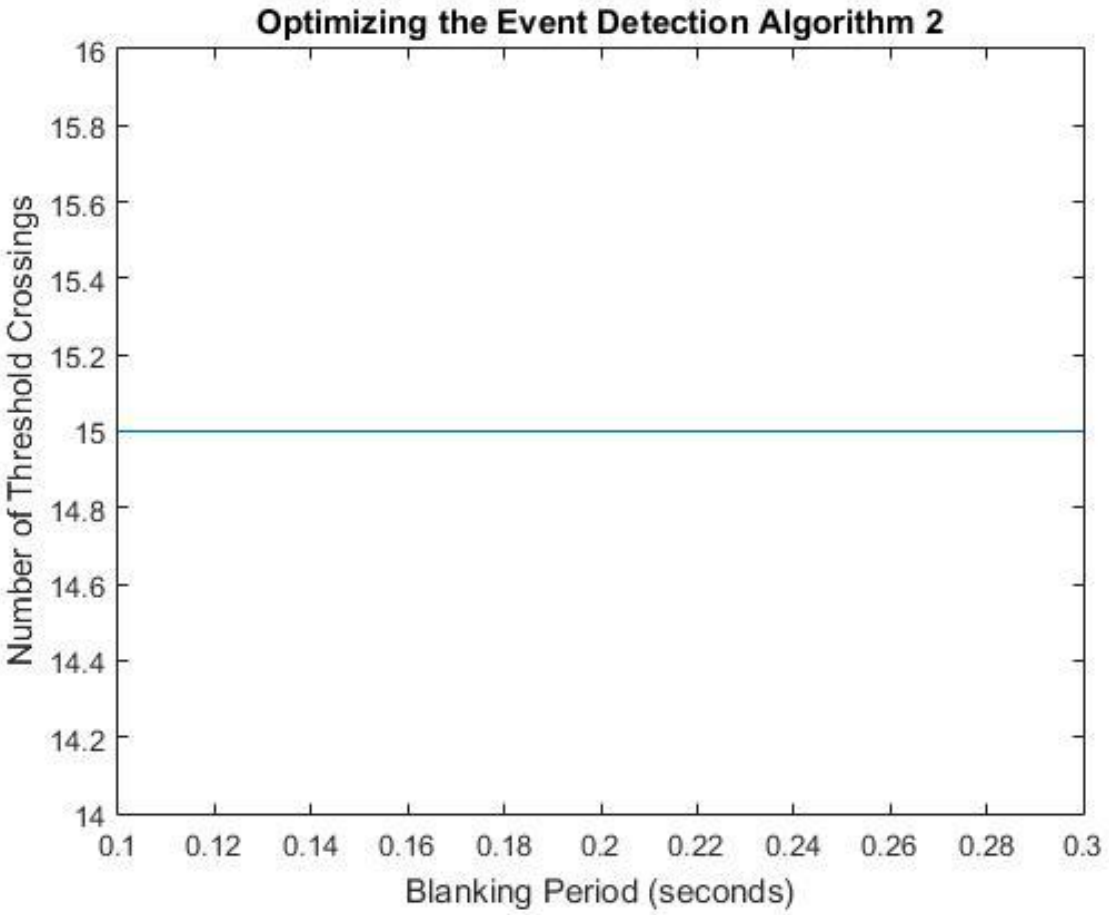


Figure 3: Plot of number of crossings for input blanking period after a threshold crossing with an amplitude threshold of 70%.

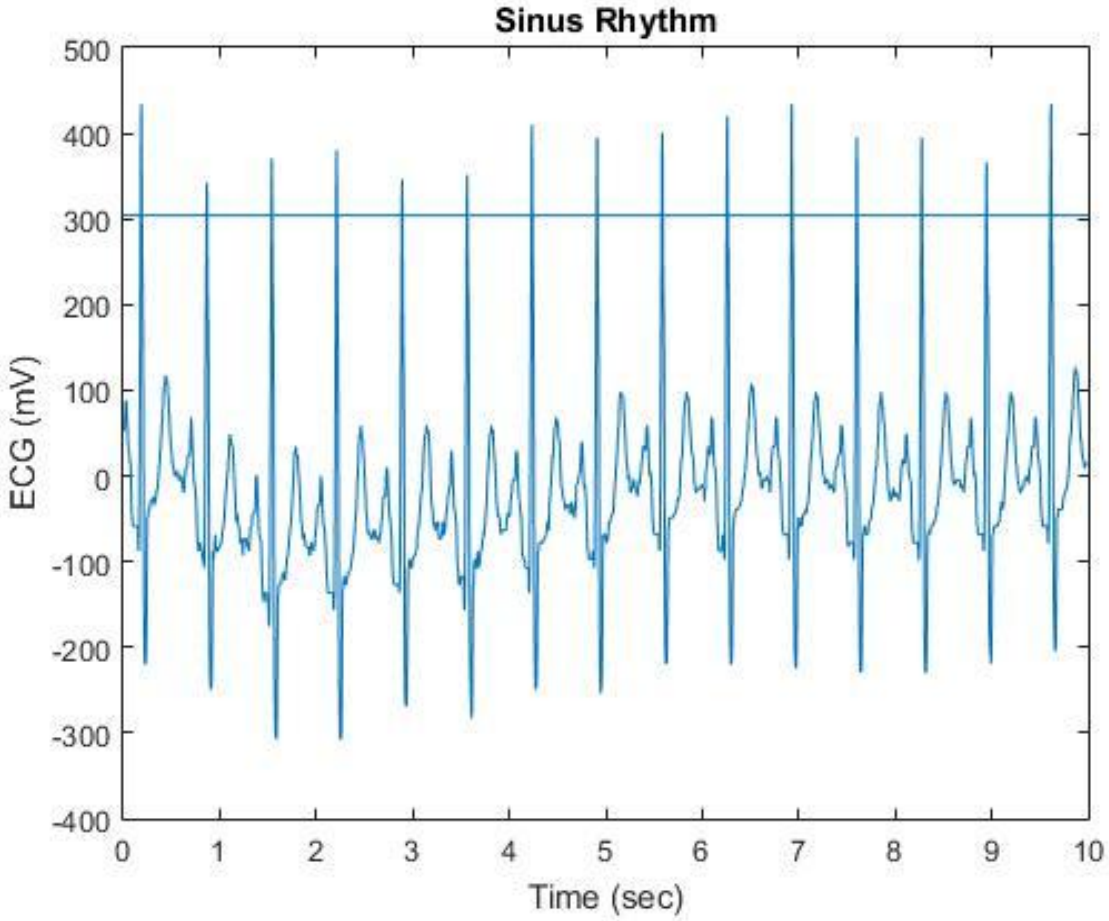


Figure 4: Plot of the data contained in the sinus4.txt file with amplitude threshold of 70% of ECG_{max} included.

Discussion:

The objectives of reaching a better understanding of signal processing for electrophysiological systems was achieved. Through the use of C functionality, such as the End Of File (EOF) statement, decision statements, pointers, loops, and dynamic memory allocation, the program was made functional and generalized. Dynamic memory allocation comes in handy when the size of the data set is unknown. Unfortunately, the program is not incredibly modular. The main function is much larger than it should be, which definitely breaks good coding practices. Using pointers as function input arguments is advantageous because the actual size of the memory for that argument does not have to be known until runtime usage. Unfortunately, this can lead to problems caused by user error in performing the task of running the program. The allocation of memory seems to be a problem for performance of the program. Perhaps there is a better way than the currently implemented process.

This algorithm has many limitations. The constant threshold value does not allow for natural physiological changes in the heart rhythm. Low-amplitude QRS complexes could cause problems for the detection algorithm, if ECG_{max} is significantly larger than those low-amplitude QRS complexes. If the blanking period is too larger, then the possibility of missing a QRS complex becomes a problem. On the other hand, a blanking period that is too small slows down the performance of the algorithm.

Conclusion:

The optimal amplitude threshold for the sample data provided is 70% of ECG_{max}. The fastest version of the algorithm would have a blanking period of 0.3 seconds for the given data set. Using the results of the algorithm and the length of time of the dataset, the average heart rate for the sampled patient was found to be just over 90 beats per minute.

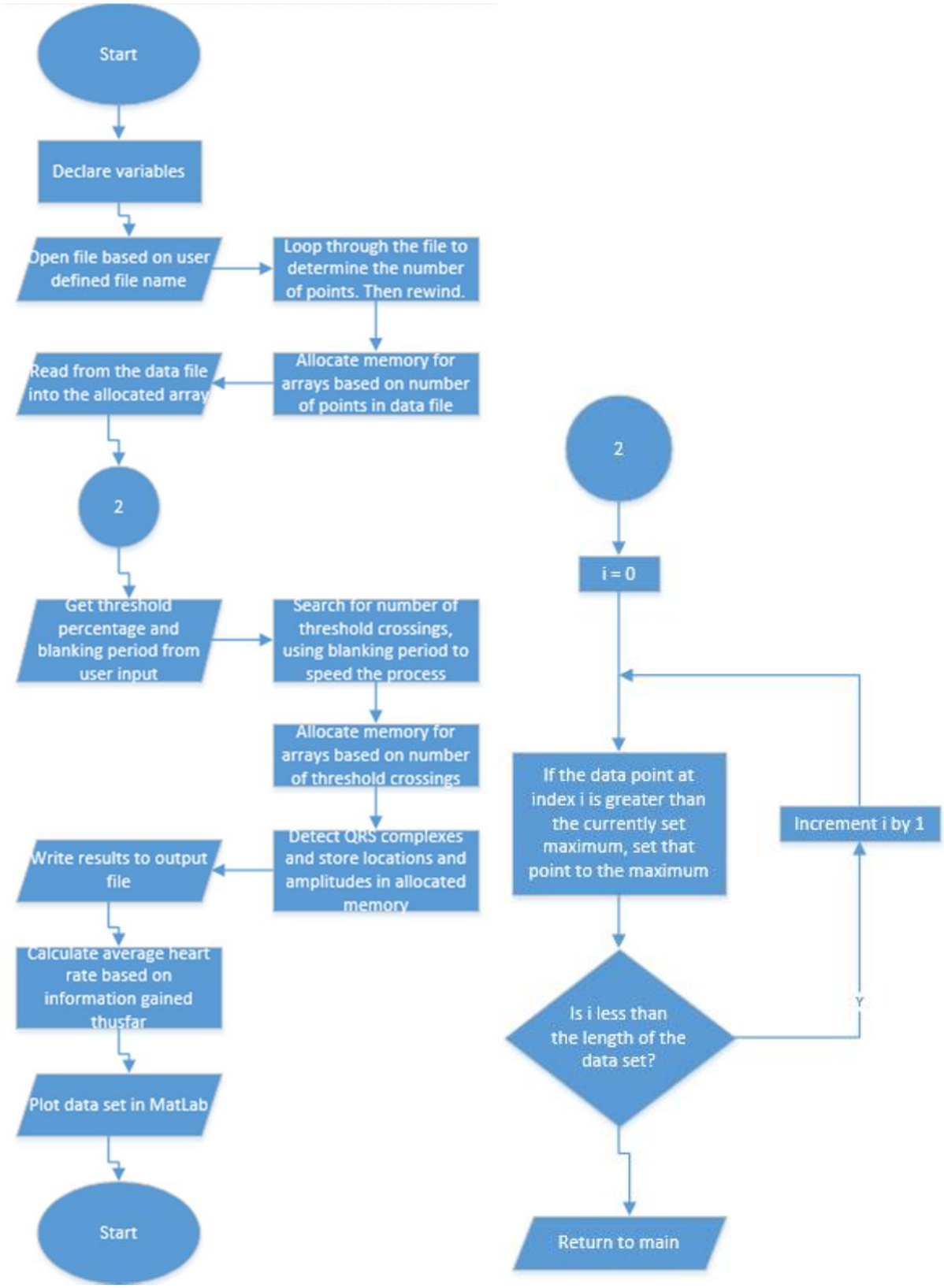
C Code:

```
/* EventDetection.cpp : Defines the entry point for the console application.
              Peter Dobbs
              BIEN 3200 - Section 401
              4 October 2016
              Lab 3

              Program Description:
                     This program implements a basic event detection algorithm to
determine
                     the event of a QRS complex. Based on the number of events in a given
                     span of time for the data set, the average heart rate is calculated.
*/

#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "engine.h"
#include "mex.h"

float threshold, maxECG, blank;
float numCrossings = 0;

void setEcgMax(float *data, int n) {
       maxECG = 0;
       for (int i = 0; i < n; ++i) { //find maxECG from input data
              if (data[i] > maxECG) maxECG = data[i];
       }
}

int main(void) {
       int n = 0;
       float *tData, *aData, t, y;
       char in_file[100];
       FILE *inputFile;

       printf("Enter input file name\n"); // open file to be read
       scanf("%s", in_file);
       inputFile = fopen(in_file, "r");

       if (!(inputFile)) {
              printf("file does not exist.\n");
              exit(EXIT_FAILURE);
       }

       // determine the number of points in the opened file
       for (n = 0; fscanf(inputFile, "%f %f", &t, &y) != EOF; ++n);
       printf("number of data points in file = %i\n", n);

       // move cursor to beginning of indicated file
       rewind(inputFile);

       tData = (float *)calloc(sizeof(float), n); //allocate memory
       aData = (float *)calloc(sizeof(float), n);
```

```
	for (int i = 0; i < n; ++i) {
		fscanf(inputFile, "%f\t%f", &tData[i], &aData[i]);
	}
	fclose(inputFile);

	//Event Detection Algorithm
	char out_file[100];
	FILE *outFile;
	float percentage, *locations, *amplitudes;
	int delT = tData[n - 1] - tData[0]; // (delT :: delta T) variable for determining
sample time

	setEcgMax(aData, n);
	printf("ECGmax = %f\n", maxECG);

	printf("Enter a threshold percentage of ECGmax\n");
	scanf("%f", &percentage);
	threshold = maxECG * percentage; //amplitude threshold for detecting QRS complex

	printf("Enter a blanking period between 0.1 and 0.3 seconds\n");
	scanf("%f", &blank);
	blank = blank * (n / delT); //specifying the blanking period for the input dataset

	for (int i = 0; i < n; ++i) {
		if (aData[i] > threshold) {
			++numCrossings;
			i = i + blank;
		}
	}
	locations = (float *)calloc(sizeof(float), numCrossings); //allocate memory
	amplitudes = (float *)calloc(sizeof(float), numCrossings);

	for (int i = 0, j = 0; i < n; ++i) { //detect QRS complexes
		if (aData[i] > threshold) {
			locations[j] = tData[i];
			amplitudes[j] = aData[i];
			++j;
			i = i + blank;
		}
	}

	printf("Enter ouput file name\n"); //write to file
	scanf("%s", out_file);
	outFile = fopen(out_file, "w");
	for (int i = 0; i < numCrossings; ++i) {
		fprintf(outFile, "%f\t%f\n", locations[i], amplitudes[i]);
	}
	fclose(outFile);
	free(locations);
	free(amplitudes);

	float avgHR = numCrossings / (tData[n - 1] / 60);
	printf("Average Heart Rate (beats/minute) = %f\n", avgHR);

	system("PAUSE");
	////////====MATLAB====//////////////////
	//Open MATLAB engine
```

```
	Engine *ep;
	ep = engOpen(NULL);

	//create pointer of type mxAray (MATLAB array) with the right dimension and type,
allocate space
	mxArray* matx1 = mxCreateNumericMatrix(n, 1, mxSINGLE_CLASS, mxREAL);
	mxArray* maty1 = mxCreateNumericMatrix(n, 1, mxSINGLE_CLASS, mxREAL);
	mxArray* thresh = mxCreateNumericMatrix(1, 1, mxSINGLE_CLASS, mxREAL);

	/*copies values of x1 and y1 into the allocated space:
		copies 10 *(#of bytes for a float) bytes from x1 to matx1 or from y1 to
maty1*/
	memcpy((void*)mxGetPr(matx1), (void*)tData, sizeof(float) * n);
	memcpy((void*)mxGetPr(maty1), (void*)aData, sizeof(float) * n);
	memcpy((void*)mxGetPr(thresh), (void*)&threshold, sizeof(float) * 1);

	/* Send variable to MatLabthrough A and F:
		Write mxArraymatx1 or maty1 to the egnine"ep"and
		gives it the variable name A or B */
	engPutVariable(ep, "A", matx1);
	engPutVariable(ep, "F", maty1);
	engPutVariable(ep, "T", thresh);

	//use Matlabcommands
	engEvalString(ep, "figure; plot(A,F);"
		"title ('Sinus Rhythm');"
		"xlabel('Time (sec)');"
		"ylabel('ECG (mV)');"
		"refline(0,T);");

	//Pauses figure
	system("pause");

	//close the engine
	engClose(ep);
	//////////======C======/////////////////

	free(tData);
	free(aData);

	system("PAUSE");
	//end program
	return 0;
}
```

Flow Charts:


Start
Declare variables
Open file based on user defined file name
Loop through the file to determine the number of points. Then rewind.
Allocate memory for arrays based on number of points in data file
Read from the data file into the allocated array
2
Get threshold percentage and blanking period from user input
Search for number of threshold crossings, using blanking period to speed the process
Allocate memory for arrays based on number of threshold crossings
Detect QRS complexes and store locations and amplitudes in allocated memory
Write results to output file
Calculate average heart rate based on information gained thusfar
Plot data set in MatLab
Start
2
i = 0
If the data point at index i is greater than the currently set maximum, set that point to the maximum
Increment i by 1
Is i less than the length of the data set?
Y
Return to main