## Table of Contents

# Task Scheduler System (C#)

Task scheduling system used to ping web a URL to allow the web system execute its code. This system uses Quartz.NET for scheduling tasks. Below are the main functions used in this system. Start(), End(), CreateJob(), ScheduleJob(), and the Jobs class.

```
private static IScheduler sc;

    /// <summary>
    /// Creates a Job and Trigger for Scheduling
    /// </summary>
    /// <param name="JobIdentity">Job Name</param>
    /// <param name="TriggerIdentity">Trigger Name</param>
    /// <param name="URL">URL to Poke</param>
    /// <param name="ScheduledTime">Input String for the Trigger Time</param>
    /// <returns>an Object that can be</returns>
    public static object CreateJob(string JobIdentity, string TriggerIdentity, string URL, string ScheduledTime)
    {
        IJobDetail job = JobBuilder.Create<Jobs>()
            .WithIdentity(JobIdentity)
            .UsingJobData("URL", URL)
            .Build();

        ITrigger trigger = TriggerBuilder.Create()
                                .ForJob(job)
                                .WithIdentity(TriggerIdentity)
                                .WithCronSchedule(ScheduledTime)
                                .Build();

        NewJob JobObj = new NewJob(job, trigger);

        return JobObj;

    }
    /// <summary>
    /// Allows a User to Schedule a Task
    /// </summary>
    /// <param name="job">Job to be scheduled</param>
    /// <param name="trigger">Trigger that starts the Scheduled Job</param>
    public static int ScheduleJob(IJobDetail job, ITrigger trigger)
    {
        try
        {
            sc.ScheduleJob(job, trigger);

            System.Diagnostics.Debug.WriteLine("Job Scheduled Successfully");
            Global.WriteToTextFile("Job Scheduled Successfully" + DateTime.Now.ToString());
            Global.WriteToTextFile(job.ToString());
            return 1;
        }
        catch (Exception ex)
        {
            System.Diagnostics.Debug.WriteLine(ex);
            Global.WriteToTextFile(ex.ToString());
            return 0;
        }

    }
    /// <summary>
    /// Starts the Scheduler on Application Load Via Global.asax
    /// </summary>
    public static void Start()
    {
        ISchedulerFactory sf = new StdSchedulerFactory();
        sc = sf.GetScheduler();
        sc.Start();

    }
    /// <summary>
    /// Ends the Scheduler
    /// </summary>
    public static void End()
    {
        if (sc.IsShutdown == false)
        {
            sc.Shutdown();
            System.Diagnostics.Debug.WriteLine("Scheduler ShutDown");
        }
    }

/// <summary>
/// Summary description for Jobs
/// </summary>
public class Jobs : IJob
{

    public void Execute(IJobExecutionContext context)
    {
        JobKey key = context.JobDetail.Key;
        JobDataMap dataMap = context.JobDetail.JobDataMap;

        string URL = dataMap.GetString("URL");

        HitRequestSite(URL);
    }

    public void HitRequestSite(string URL)
    {
        WebClient client = new WebClient();
        client.DownloadData(URL);
    }

}
```

# Stock Trader Engine (C#)

This is the trader engine for a stock trading game that I am currently working on. This game is going to use TCP connection for the clients to connect to the server and execute trades. The code that I have complete is the trader engine, which is reliant on a SQL Server for all the trade executions. I have created functions for selling, buying, and adjusting the market stocks.

```
/// <summary>
    /// Checks to make sure there is enough stock to buy
    /// </summary>
    /// <param name="StockID"></param>
    /// <param name="AmountToBuy"></param>
    /// <returns></returns>
    public bool CheckStockAmount(int StockID, int AmountToBuy)
    {
        SqlConnection conn = glob.Connect();
        conn.Open();
        string sql = "select Amount FROM CURRENT_STOCK_PRICES as CS where CS.StockID = @StockID ";
        SqlCommand cmd = new SqlCommand(sql, conn);
        cmd.Parameters.Add(new SqlParameter("StockID", StockID));
        int AmountAvaliable = (int) cmd.ExecuteScalar();
        if (AmountAvaliable >= AmountToBuy)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public float GetCurrentStockPrice(int StockID)
    {
        float CurrentStockPrice = 0;
        SqlConnection conn = glob.Connect();
        conn.Open();
        string sql = "select CS.StockPrice FROM CURRENT_STOCK_PRICES CS where CS.StockID=@StockID ";
        SqlCommand cmd = new SqlCommand(sql, conn);
        cmd.Parameters.Add(new SqlParameter("StockID", StockID));
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            CurrentStockPrice = (float)(double)dr["StockPrice"];
        }
        dr.Close();
        glob.CloseDB(conn);
        return CurrentStockPrice;
    }

    public float AdjustStockPrice(int StockID, int UserID,int Amount, int Action, int StockTransLog)
    {
        string Now = DateTime.Now.ToString();
        string AnHourAgo = DateTime.Now.AddHours(-1).ToString();

        int SellsCount = 0;
        int BuysCount = 0;
```

```
            double Multiplier = .0035;

            SqlConnection conn = glob.Connect();
            conn.Open();
            string sql = "select AmountBefore, AmountAfter FROM STOCK_TRANS_LOG Where StockID=@StockID and 
[Date]>=@HourAgo and [Date]<=@Now";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("HourAgo", AnHourAgo));
            cmd.Parameters.Add(new SqlParameter("Now", Now));
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                if ((int)dr["AmountBefore"] > (int)dr["AmountAfter"])
                {
                    SellsCount++;
                }
                else
                {
                    BuysCount++;
                }
            }
            dr.Close();

            if (SellsCount > BuysCount)
            {
                Multiplier *= -1;
            }
            else if (SellsCount == BuysCount)
            {
                Multiplier = 1;
            }

            //Check to see if the last transaction was a buy or sell

            //Then recalculate the Stock value

            //Then recalculate with the multiplier

            //Insert the new stock price into the DB

            glob.CloseDB(conn);

            return 0;
        }

        /// <summary>
        /// Sells Stock for a User and adds the Transactions to the Logs
        /// </summary>
        /// <param name="UserID">Users Submitting the Request</param>
        /// <param name="StockID">Stock Selling</param>
        /// <param name="Amount">Amount of Stock to sell</param>
        /// <returns></returns>
        public bool SellStock(int UserID, int StockID, int Amount)
        {
            float CurrentStockPrice = GetCurrentStockPrice(StockID);
            float Cost = CurrentStockPrice * Amount;
            SqlConnection conn = glob.Connect();
            conn.Open();

            //Removes the stock to the Users Account
            string sql = "update USER_STOCKS set Amount=(Amount - @Amount), [Date]=@Date where StockID=@StockID
and UserID=@UserID ";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("Amount", Amount));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();


            //Removes the bought stock from the market
            sql = "update USER_CASH set Cash=(Cash + @Cash), [Date]=@Date where UserID=@UserID ";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("Cash", Cost));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();


            //Add the Transaction to the Log
            sql = "Insert into USER_TRANS_LOG (UserID, StockID, Method, Amount, Cost, [Date]) Values (@UserID,
@StockID, @Method, @Amount, @Cost, @Date)";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("Method", "0"));
            //Need to change this to the recalculated value
            cmd.Parameters.Add(new SqlParameter("Amount", Amount));
            cmd.Parameters.Add(new SqlParameter("Cost", Cost));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();

            int AmountAfterSell = 0;



            //Adds the bought stock from the market
            sql = "update CURRENT_STOCK_PRICES set Amount=(Amount + @Amount) output inserted.Amount,
inserted.StockPrice where StockID=@StockID ";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("Amount", Amount));
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                AmountAfterSell = (int) dr["Amount"];
            }
            dr.Close();

            //Readjusts the Stock Price
            //Have not come up with the equation to adjust the price yet

            //Add the Transaction to the Log
            sql = "insert into STOCK_TRANS_LOG  (UserID, StockID, BeforeChange, AfterChange, AmountBefore,
AmountAfter, [Date]) Values (@UserID, @StockID, @MoneyBefore, @MoneyAfter, @AmountBefore, @AmountAfter,
@Date)";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("MoneyBefore", CurrentStockPrice));
            //Need to change this to the recalculated value
            cmd.Parameters.Add(new SqlParameter("MoneyAfter", CurrentStockPrice));
            cmd.Parameters.Add(new SqlParameter("AmountBefore", (AmountAfterSell - Amount)));
            cmd.Parameters.Add(new SqlParameter("AmountAfter", AmountAfterSell));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();

            glob.CloseDB(conn);
            return true;
        }


        /// <summary>
        /// Buys Stock on the market
        /// </summary>
        /// <param name="UserID"></param>
        /// <param name="StockID"></param>
        /// <param name="Amount"></param>
        /// <returns></returns>
        public bool BuyStock(int UserID, int StockID, int Amount)
        {
            float CurrentStockPrice = 0;
            float Cash = 0;
            SqlConnection conn = glob.Connect();
            conn.Open();
            string sql = "select CS.StockPrice FROM CURRENT_STOCK_PRICES CS where CS.StockID=@StockID ";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                CurrentStockPrice = (float) (double) dr["StockPrice"];
            }
            dr.Close();

            sql = "select Cash FROM [USER] as U inner join USER_CASH as UC on UC.UserID = U.UserID where
    U.UserID=@UserID";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                Cash = (float) (double) dr["Cash"];
            }
            dr.Close();

            if ((CurrentStockPrice * Amount <= Cash) && CurrentStockPrice != 0)
            {
               bool StockBough =  AdjustMarketStock(StockID, UserID, Amount);
               bool UserUpdated = AdjustUser(UserID, (CurrentStockPrice * Amount), Amount, StockID);
                if (StockBough == true && UserUpdated == true)
                {
                    return true;
                }
            }

            glob.CloseDB(conn);
            return false;
        }

        /// <summary>
        /// Buys the Stock for a User
        /// </summary>
        /// <param name="StockID">Stock to be purchased</param>
        /// <param name="UserID">User Buying the Stock</param>
        /// <param name="StockBought">Amount of Stock to buy</param>
        /// <returns></returns>
        public bool AdjustMarketStock(int StockID, int UserID, float StockBought)
        {
            int AmountAfterBuy = 0;
            float StockPrice = 0;

            SqlConnection conn = glob.Connect();
            conn.Open();
            //Removes the bought stock from the market
            string sql = "update CURRENT_STOCK_PRICES set Amount=(Amount - @Amount) output inserted.Amount,
inserted.StockPrice where StockID=@StockID ";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("Amount", StockBought));
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                AmountAfterBuy = (int)dr["Amount"];
                StockPrice = (float) (double) dr["StockPrice"];
            }
            dr.Close();

            //Readjusts the Stock Price
            //Have not come up with the equation to adjust the price yet

            //Add the Transaction to the Log
            sql = "insert into STOCK_TRANS_LOG  (UserID, StockID, BeforeChange, AfterChange, AmountBefore,
AmountAfter, [Date]) Values (@UserID, @StockID, @MoneyBefore, @MoneyAfter, @AmountBefore, @AmountAfter,
@Date)";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("MoneyBefore", StockPrice));
            //Need to change this to the recalculated value
            cmd.Parameters.Add(new SqlParameter("MoneyAfter", StockPrice));
            cmd.Parameters.Add(new SqlParameter("AmountBefore", (AmountAfterBuy + StockBought)));
            cmd.Parameters.Add(new SqlParameter("AmountAfter", AmountAfterBuy));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();

            glob.CloseDB(conn);
            return true;

        }

        /// <summary>
        /// Adds the stock to the Users Account
        /// </summary>
        /// <param name="UserID"></param>
        /// <param name="Cost"></param>
        /// <param name="Amount"></param>
        /// <param name="StockID"></param>
        /// <returns></returns>
        public bool AdjustUser(int UserID, float Cost, int Amount, int StockID)
        {
            SqlConnection conn = glob.Connect();
            conn.Open();
            //Removes the bought stock from the market
            string sql = "update USER_CASH set Cash=(Cash - @Cash), [Date]=@Date where UserID=@UserID ";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("Cash", Cost));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();

            //Adds the stock to the Users Account
            sql = "update USER_STOCKS set Amount=(Amount + @Amount), [Date]=@Date where StockID=@StockID and 
UserID=@UserID ";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("Amount", Amount));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();


            //Add the Transaction to the Log
            sql = "Insert into USER_TRANS_LOG (UserID, StockID, Method, Amount, Cost, [Date]) Values (@UserID, 
@StockID, @Method, @Amount, @Cost, @Date)";
            cmd = new SqlCommand(sql, conn);
            cmd.Parameters.Add(new SqlParameter("StockID", StockID));
            cmd.Parameters.Add(new SqlParameter("UserID", UserID));
            cmd.Parameters.Add(new SqlParameter("Method", 1));
            //Need to change this to the recalculated value
            cmd.Parameters.Add(new SqlParameter("Amount", Amount));
            cmd.Parameters.Add(new SqlParameter("Cost", Cost));
            cmd.Parameters.Add(new SqlParameter("Date", DateTime.Now.ToString()));
            cmd.ExecuteNonQuery();

            return true;
        }
    }
}
```

## Image Conversion (C#)

These are functions that can convert images to base64 strings and back. Also, I created a function for an internship that converts byte strings into images again. I created this function to convert all our photos stored on an old system’s database in byte strings.

```
static void Main(string[] args)
    {
      string ImagePath = Path.Combine("H:\\Work Space\\Base64Converter", "table.png");
      string basePath = "H:\\Work Space\\Base64Converter";
      Program prog = new Program();

      string ReturnBaseStr = prog.ImageToBase64(ImagePath);

      Console.Out.WriteLine(ReturnBaseStr);

      Image img = prog.Base64ToImage(ReturnBaseStr);

      string newImgPath = Path.Combine(basePath, "newtable2.jpeg");
      img.Save(newImgPath);

      Console.Out.WriteLine(newImgPath);


      string HexString = File.ReadAllText(Path.Combine(basePath, "HexFile.txt"));
      int i = 2;
      prog.hexToByteArray(HexString, "NewImage.jpeg");

    }


    public string ImageToBase64(string Path)
    {
      using (Image image = Image.FromFile(Path))
      {
        using (MemoryStream m = new MemoryStream())
        {
          image.Save(m, image.RawFormat);
          byte[] imageBytes = m.ToArray();
          // Convert byte[] to Base64 String
          string base64String = Convert.ToBase64String(imageBytes);
          return base64String;
        }
      }
    }

    public Image Base64ToImage(string base64String)
    {
      // Convert Base64 String to byte[]
      byte[] imageBytes = Convert.FromBase64String(base64String);
      MemoryStream ms = new MemoryStream(imageBytes, 0,
        imageBytes.Length);

      // Convert byte[] to Image
      ms.Write(imageBytes, 0, imageBytes.Length);
      Image image = Image.FromStream(ms, true);
      return image;
    }
```

```
        private void hexToByteArray(string hexString, string FileName)
{
    int bytesCount = (hexString.Length) / 2;
    byte[] bytes = new byte[bytesCount];
    for (int x = 0; x < bytesCount; ++x)
    {
        bytes[x] = Convert.ToByte(hexString.Substring(x * 2, 2), 16);
    }

    MemoryStream mm = new MemoryStream(bytes);
    Image _image = System.Drawing.Image.FromStream(mm);

    string basePath = "H:\\Work Space\\Base64Converter";
    _image.Save(Path.Combine(basePath,FileName));
}
```

## Password Hasher with Salt (C#)

This is a windows for application that I created to test hashing strings with random salt. I also added a way to check the hashes against each other.

```
 public string Salt = "";
        public Form1()
        {
            InitializeComponent();
        }

        private void Hashbtn_Click(object sender, EventArgs e)
        {
            Salt = CreateRandomSalt();
            HashedPass.Text = Hash(PlainPass.Text, Salt);
        }

        static string CreateRandomSalt()
        {
            string mix =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()_+=][}{<>";
            string salt = "";
            Random rnd = new Random();
            StringBuilder sb = new StringBuilder();
            for (int i = 1; i <100; i++)
            {
                int x = rnd.Next(0, mix.Length - 1);
                salt += (mix.Substring(x, 1));
            }
            return salt;
        }

        static string Hash(string input, string salt)
        {
            Byte[] convertedToBytes = Encoding.UTF8.GetBytes(input + salt);
            HashAlgorithm hashType = new SHA512Managed();
            Byte[] hashBytes = hashType.ComputeHash(convertedToBytes);
            string hashedResults = Convert.ToBase64String(hashBytes);


            return hashedResults;
        }

        private void Checkbtn_Click(object sender, EventArgs e)
        {
            if (HashedPass.Text != "")
            {
                string CheckedHash = Hash(Checktxt.Text, Salt);

                bool result = HashedPass.Text.ToString().Equals(CheckedHash, StringComparison.Ordinal);
                if (result == true )
                {
                    Checklb.Text = "Match!";
                    Checklb.ForeColor = Color.Green;
                    Checklb.Visible = true;

                }
                else
        {
            Checklb.Text = "Wrong!";
            Checklb.ForeColor = Color.Red;
            Checklb.Visible = true;
        }
    }
}
```

## Multi Process Word Counter (C)

This is a group project I worked on that splits an input text file into n processes with n -1 being children and 1 parent. Each process counts the number of words in its section of the file then uses pipes to send the counted words back to the parent process. After all of the children return their data to the parent process, the word counts are sorted and written to an output file.

```
/**
 **  Cmpsc 311 -  Project 2 - multiprocess word count
 ** Team - Tyler Lutz & Dylan Steele
 **
 ** Program works by reading from input file, removing special character and writing the result to a temp file
 ** It then reads each word from the temp file into an element of a char* array and uses the qsort() function to sort it
 ** The parent process then creates n-1 children and each process parses part of the array of words and obtains a word
count
 ** Then the children pipe their results back to the parent process which concatenates the results into a final word count
 ** Finally the word count is printed to the output file and the run_time is recorded
 **
 */


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>

#define MAX 1024

typedef int bool;
enum { false, true };

struct word
{
		char* word;
		int count;
		struct word* next;
};


//---------------------------------
//--Function for qsort comparison--
//---------------------------------
int cstring_cmp(const void *a, const void *b)
{
    const char **ia = (const char **)a;
    const char **ib = (const char **)b;
    return strcmp(*ia, *ib);
	/* strcmp functions works exactly as expected from
	comparison function */
}


int main(int argc, char* argv[]) {
	//create clock_t varibables to record run-time
	clock_t begin, end;
```

```
        //double to hold value of run-time
        double run_time;
        //set beginning time
        begin = clock();

#ifdef DEBUG
        printf("\n--The program is starting--\n\n");
#endif



        //create file objects for opening file
        FILE *ifp, *ofp;
        //create strings holding location of the intput and output files
        char *file, *temp, *out, *run_ti;
                //create int to hold number of children to use
                int n;

        if(argc == 2) {
                 file = argv[1];
                 out = "word_count.txt";
                 run_ti = "run_time.txt";
                n = 2;
                }
        else if(argc == 3) {
                 file = argv[1];
                 out = "word_count.txt";//argv[2];
                 run_ti = "run_time.txt";
                                n = atoi(argv[2]);
        }
        else if(argc == 4) {
                 file = argv[1];
                 out = argv[2];
                 run_ti = argv[3];
                                n = 2;
        }
                else if(argc == 5) {
                                file = argv[1];
                                out = argv[2];
                                run_ti = argv[3];
                                n = atoi(argv[4]);
                }
        else {//solve errors
                 printf("Enter the name of a local .txt file to run word count on: ");
                 scanf("%s", file);
                 printf("Enter the name of the output file to write word count to: ");
                 scanf("%s", out);
                 printf("Enter the name of the output file to write the run-time to: ");
                 scanf("%s", run_ti);
                                printf("Enter the number of children processes to use: ");
                                scanf("%d", n);
        }
        temp = "temp.txt";

        char buff[255];

        //open the file streams to read and write to the files
        ifp = fopen(file, "r");
        ofp = fopen(temp, "w+");

#ifdef DEBUG
```

```
	printf("--File Streams opened--\n\n--Now removing special chaacters and making all letters lowercase--\n\n");
#endif

	//remove any special characters and turn all capitals to lower case

	int c;
	while((c = getc(ifp)) != EOF)
		if(isalpha(c) || isspace(c) || isdigit(c))
			fputc(tolower(c), ofp);

	fclose(ofp);//close stream to open with new permission

	ofp = fopen(temp, "r");//reopen stream


	bool null_ele;
	int size = 0;

#ifdef DEBUG
	printf("--Special characters removed and all letters now lowercase--\n - now reading words into unsorted array -\n\n");
#endif

	//get the size of the file to be read in
	while(fscanf(ofp, "%s", buff) != EOF)   size++;
	//create array to allow sorting of words
	char** raw_words = malloc(size*sizeof(char*));
	//create iterator int to populate values of raw_words
	int i = 0;

	fclose(ofp);//close stream to open with new permission
	ofp = fopen(temp, "r");//reopen stream

	//copy each word from the file into an element of the array

	while(fscanf(ofp, "%s", buff) != EOF)
		raw_words[i++] = strdup(buff);


#ifdef DEBUG
	printf("--All words read into array--\n - now sorting the array of words -\n\n");
#endif

	//sort array of words
	qsort(raw_words, size, sizeof(char*), cstring_cmp);

#ifdef DEBUG
	printf("--Array is now sorted--\n - now creating n-1 children and corresponding pipes -\n\n");
#endif

		int pipes[n-1];//array to hold file decriptors for read end of n-1 pipes
		int fd[2];//file descriptor for creating n-1 pipes


		pid_t pid;
		pid = getpid();


		int start_index, range, remain;//create variables for the start_index and range of each process
		remain = size%n;//if the size is odd find the remainder
		range = size/n;//set the range to size*(1/n) so each child works on only their range
```

```
                //loop n-1 times starting with index i = 1
                for(i = 1; i < n; i++){
                  //set start_index to range * current index + remainder offset (counted by parent)
                  start_index = remain + (range)*i;

                  //create a pipe for communicating between child and parent
                  if((pipe(fd)) < 0){
                                //failure in creating pipe
                                perror("pipe error\n");
                                exit(1);
                  }

                  // keep track of the read end of this pipe so parent can read from each pipe created
                  pipes[i-1] = fd[0];

                  //fork and create child associated to current start_index and fd (alowing it to later write to fd[1])
                  if ((pid = fork()) < 0) {
                      //failure in creating a child
                perror ("fork error\n");
                       exit(2);
                   }

                  //if child
            if(pid == 0){
                      close(fd[0]);//close read end
                break;
                  }
                  else
                                close(fd[1]);//otherwise close write end

                }

#ifdef DEBUG
        printf("--Children and pipes created--\n - begining to count through sorted array concurrently -\n\n");
#endif

                //create list pointers to begin counting
                int j, count, end_range;
        struct word *head, *curr, *temp_pt;
        curr = head = (struct word*)malloc(sizeof(struct word));
                //if the parent
                if(pid != 0)
                {
                                start_index = 0;//set start_index to 0 so it does the first part of array
                                range += remain;//add remainder to range of parent so no words are missed
                }
                //set the end of range for each process
        end_range = start_index + range;

        //for each process start at the given start_index and stop at end_range
        for(i = start_index; i < end_range; i++)
        {
                        count = 1;
                for(j = i+1; j < end_range; j++)
                {
                                                if(strcmp(raw_words[i], raw_words[j]) == 0)
                        {
                                        count++;
                                free(raw_words[j]);
                        }
                        else
                        break;
                }


               temp_pt = (struct word*)malloc(sizeof(struct word));

                        temp_pt->word = strdup(raw_words[i]);
               temp_pt->count = count;
               temp_pt->next = NULL;
               curr->next = temp_pt;
                        free(raw_words[i]);

                        i = j-1;
                        curr = curr->next;
        }


#ifdef DEBUG
        printf("--Process %ld has finished it's word count--\n\n", (long)getpid());
#endif

               if(pid == 0 )
               {
                        //is child

                        #ifdef DEBUG
                                printf("--Child process \"%ld\" has begun piping data--\n\n", (long)getpid());
                        #endif

                        //THIS is the odd part - needed to set curr = head->next, if set to head the first value
                        //is "a" and has an obscene count# (e.g. 6356214)
                        //making this change in the parent and child when traversing the list seems to solve the issue
                        curr = head->next;
                        char count[MAX];

                        while(curr != NULL)
                        {
                                char message[strlen(curr->word)+1];

                                //format strings before writing to pipe in order to use fgets() function
                                sprintf(message, "%s\n", curr->word);
                                sprintf(count, "%d\n", curr->count);

                                write(fd[1], count, (strlen(count)));
                                write(fd[1], message, (strlen(message)));

                                curr = curr->next;
                        }
                        //child has gone through it's part of the array
                        close(fd[1]);

                        #ifdef DEBUG
                                printf("--Child process \"%ld\" has finished piping data--\n\n", (long)getpid());
                        #endif

                        exit(1);
               }
               else{
                        //is parent

                        #ifdef DEBUG
```

```
			printf("--Parent process \"%ld\" is waiting for it's children to finish--\n\n", (long)getpid());
		#endif

		//wait for all child processes to write to their pipes
		wait(NULL);

		#ifdef DEBUG
			printf("--Parent process \"%ld\" has begun reading piped data from it's children--\n\n",
(long)getpid());
		#endif

		char w_count[MAX], word[MAX];
		int wcount;
		FILE *stream;
		//for each child
		for(i = 0; i < n-1; i++)
		{
			//open the pipe from that given child
			stream = fdopen(pipes[i], "r");

			//continuously loop until a break condition
			while (1)
			{
				//Clearing the message buffer
	memset (w_count, 0, sizeof(w_count));
				memset (word, 0, sizeof(word));
			//Reading message from the pipe

				//Read until there is nothing left in the pipe
				if(fgets(w_count, MAX, stream)==NULL)
					break;
				if(fgets(word, MAX, stream)==NULL)
					break;


				//below snippet was pulled from stackoverflow and replaces '\n' with '\0'
				char *pos;
				if((pos = strchr(word, '\n')) != NULL)
					*pos = '\0';

				wcount = atoi(w_count);

				curr = head->next;


				//now search through the parent linked list created during its word counting
				//and add to list if not already there

				int comp;

				while(curr != NULL)
				{
					//Prevents error from happening in strcmp
					if(curr->count == 0)
	  break;
					else
						comp = strcmp(curr->word, word);

					//if the words are the same increment the word's count
					if(comp == 0)
					{
```

```
                                                                      curr->count+=wcount;
                                                                      break;
                                                              }
                                                              //add the a new word node in front of current - this is to keep alphabetical
order
                                                              else if(comp > 0)
                                                              {
                                                                      temp_pt = (struct word*)malloc(sizeof(struct word));
                                                                      temp_pt->word = strdup(word);
                                                                      temp_pt->count = wcount;
                                                                      temp_pt->next = curr;
                                                                      curr = temp_pt;
                                                                      break;
                                                              }
                                                              //if at the end of the list, add the new word node to the end
                                                              else if(curr->next == NULL)
                                                              {
                                                                      temp_pt = (struct word*)malloc(sizeof(struct word));
                                              temp_pt->word = strdup(word);
                                              temp_pt->count = wcount;
                                              temp_pt->next = NULL;
                                              curr->next = temp_pt;
                                                                      break;

                                                              }
                                                              curr = curr->next;
                                              }
                                             }
                                             //close stream and read end of current pipe when done with them
                                             close(pipes[i]);
                                             close(stream);
                              }

                              #ifdef DEBUG
                                             printf("--Parent process \"%ld\" has finished reading all piped data--\n - now printing final word
count - \n\n", (long)getpid());
                              #endif

                              curr = head->next;
              //open output file
              FILE *fp = fopen(out, "w+");

              while(curr != NULL)
              {
                     fprintf(fp, "%s, %d\n", curr->word, curr->count);
                     curr = curr->next;
              }
              fclose(fp);
              //printf("List printed to %s", out);

              //free the memory
              curr = head;
              while(curr != NULL)
              {
                     curr = head->next;
                     free(head);
                     head = curr;
              }

                              //close file pipes
                              close(ifp);
```

```
		close(ofp);

		free(raw_words);

		//set end of run-time
	end = clock();
	//calculate run-time
	run_time = (double)(end - begin) / CLOCKS_PER_SEC;

		#ifdef DEBUG
			printf("--Program completed execution in %G seconds--\n\n", run_time);
		#endif

		ofp = fopen(run_ti, "a");
	fprintf(ofp, "--Program completed execution using %d processes on file: %s in %G seconds\n",n, file, run_time);
	fclose(ofp);

	}
	return 0;
}
```

# Mailing Labels (VB.NET)

This is a function that creates mailing labels using the DocX library to write to word documents. It uses a mailing label word template to write to, then saves the final document as a new file.

```
''' <summary>
    ''' Creates a Word Document of Mailing Labels and Emails it to the User
    ''' </summary>
    ''' <param name="AgencyID"></param>
    ''' <returns>Returns True if Successful, False otherwise</returns>
    ''' <remarks>Created by Dylan Steele 11/9/2016</remarks>
    Public Shared Function CreateMailingLabels(ByVal AgencyID As Integer) As Boolean
        'Location of the Template that is used to create the mailing labels
        Dim TemplateLocation As String = ConfigurationManager.AppSettings("FileRoot") & "documents\Templates"
        Dim DocumentName As String = ""

        Using conn As New SqlConnection(ConfigurationManager.ConnectionStrings("FACS").ToString)
            Dim sql As String = ""
            conn.Open()
            sql = "select (BM.FName + ' ' + BM.LName) as Name, BM.Address, BM.Address2, BM.City, BM.State, BM.Zip FRO
M BOARD_MEMBER as BM where BM.AgencyID=@AgencyID"
            Dim cmd As New SqlCommand(sql, conn)
            cmd.Parameters.Add(New SqlParameter("@AgencyID", AgencyID))
            Dim dr As SqlDataReader = cmd.ExecuteReader()

            If dr.Read() Then

                'Loads the Template
                Dim doc = DocX.Load(TemplateLocation & "\LabelsTemplate.docx")
                Dim Counter As Integer = 1

                'Loops through the DB results
                While dr.Read()
                    'Checks to make sure their is an address
                    If (IsDBNull(dr("Address")) = False) Then
                        Dim LabelString As String = dr("Name") & Environment.NewLine & dr("Address") & Environment.NewLine
                        If IsDBNull(dr("Address2")) = False Then
                            LabelString = LabelString & dr("Address2") & Environment.NewLine
                        End If
                        LabelString = LabelString & dr("City") & ", " & dr("State") & " " & dr("Zip")

                        'Inserts the Information into the Document
                        If Counter <= 30 Then
                            doc.InsertAtBookmark(LabelString, "Cell" & Counter)
                        End If

                        Counter = Counter + 1
                    End If
                End While
                dr.Close()
                'Saves the Document
                DocumentName = TemplateLocation & "\Labels" & Date.Now.Day & "_" & Date.Now.Month & "_" & Date.Now.Hou
r & "_" & Date.Now.Minute & ".docx"
                doc.SaveAs(DocumentName)

                'Sends the file through email
                Dim body As String = "Attached is the requested Mailing Labels."
                MailHelper.SendMailMessage(HttpContext.Current.Session("Email"), "", True, "IT@jccap.org", "Mailing Labels", b
ody, DocumentName, 1, Nothing)

                Return True
            Else
                Return False
            End If
            Global_Functions.CloseDB(conn)
        End Using
    End Function
    ''' <summary>
    ''' Creates a Word Document of Mailing Labels and Emails it to the User
    ''' </summary>
    ''' <param name="AgencyID"></param>
    ''' <param name="Email">Email of the Recipient</param>
    ''' <returns>Returns True if Successful, False otherwise</returns>
    ''' <remarks>Created by Dylan Steele on 11/9/2016</remarks>
    Public Shared Function CreateMailingLabels(ByVal AgencyID As Integer, ByVal Email As String) As Boolean
        'Location of the Template that is used to create the mailing labels
        Dim TemplateLocation As String = ConfigurationManager.AppSettings("FileRoot") & "documents\Templates"
        Dim DocumentName As String = ""

        Using conn As New SqlConnection(ConfigurationManager.ConnectionStrings("FACS").ToString)
            Dim sql As String = ""
            conn.Open()
            sql = "select (BM.FName + ' ' + BM.LName) as Name, BM.Address, BM.Address2, BM.City, BM.State, BM.Zip FRO
M BOARD_MEMBER as BM where BM.AgencyID=@AgencyID"
            Dim cmd As New SqlCommand(sql, conn)
            cmd.Parameters.Add(New SqlParameter("@AgencyID", AgencyID))
            Dim dr As SqlDataReader = cmd.ExecuteReader()

            If dr.Read() Then

                'Loads the Template
                Dim doc = DocX.Load(TemplateLocation & "\LabelsTemplate.docx")
                Dim Counter As Integer = 1

                'Loops through the DB results
                While dr.Read()
                    'Checks to make sure their is an address
                    If (IsDBNull(dr("Address")) = False) Then
                        Dim LabelString As String = dr("Name") & Environment.NewLine & dr("Address") & Environment.NewLine
                        If IsDBNull(dr("Address2")) = False Then
                            LabelString = LabelString & dr("Address2") & Environment.NewLine
                        End If
                        LabelString = LabelString & dr("City") & ", " & dr("State") & " " & dr("Zip")

                        'Inserts the Information into the Document
                        If Counter <= 30 Then
                            doc.InsertAtBookmark(LabelString, "Cell" & Counter)
                        End If

                        Counter = Counter + 1
                    End If
                End While
                dr.Close()
                'Saves the Document
                DocumentName = TemplateLocation & "\Labels" & Date.Now.Day & "_" & Date.Now.Month & "_" & Date.Now.Hou
r & "_" & Date.Now.Minute & ".docx"
                doc.SaveAs(DocumentName)
                'Sends the file through email
                Dim body As String = "Attached is the requested Mailing Labels."
                MailHelper.SendMailMessage(Email, "", True, "IT@jccap.org", "Mailing Labels", body, DocumentName, 1, Nothing
)

                Return True
            Else
                Return False
            End If
            Global_Functions.CloseDB(conn)
        End Using
    End Function
```