Lab 1 – Basic C

1. Combine two monaural audio signals into a stereo signal (use f1.wav and f2.wav)
   1. MATLAB: read in two monaural audio files (input media files) and write each out to a raw binary file (C input file)

```
>> audio2bin('f1.wav');
Input  file = f1.wav
Output file = f1.bin
>> audio2bin('f2.wav');
Input  file = f2.wav
Output file = f2.bin
```

   2. C: read in the headers for input raw binary files
   3. C: write out header for output raw binary file

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define IOBUFFSIZE 1024

typedef struct
{
	int ndim;	/* number of dimensions; audio=1 */
	int nchan;	/* number of channels; monaural=1 stereo=2 */
	int d0;		/* length of first dimension; audio=L L=length */
	int d1;		/* length of second dimension; audio=Fs Fs=sample rate */
	int d2;		/* length of third dimension; audio=null */
} dsp_file_header;

int main(int argc, char *argv[])
{
	/* binary file read/write */
	FILE *I1, *I2, *O; /* I1=input 1 I2=input 2 O=output */
	if (NULL == (I1 = fopen("f1.bin","rb")))
	{
		printf("ERROR: Can't open f1.bin for input.\n");
		return 0;
	}
	if (NULL == (I2 = fopen("f2.bin","rb")))
	{
		printf("ERROR: Can't open f2.bin for input.\n");
		return 0;
	}
	if (NULL == (O = fopen("f.bin","wb")))
	{
		printf("ERROR: Can't open f.bin for output.\n");
		return 0;
	}

	/* header read/write */
	dsp_file_header h1, h2, h; /* h1=header input 1 h2=header input 2 h=header output */
	fread(&h1,sizeof(dsp_file_header),1,I1);
	fread(&h2,sizeof(dsp_file_header),1,I2);
```

```
	memcpy(&h,&h1,sizeof(dsp_file_header));
	h.nchan = 2; /* output file; monaural=1 stereo=2 */
	fwrite(&h,sizeof(dsp_file_header),1,O);
/* Continue to 4. & 5. */
```

4. C: statically allocate two buffers to hold input monaural audio data
5. C: statically allocate one buffer to hold output stereo audio data

```
	/* monaural to stereo */
	float x1[IOBUFFSIZE];		/* input memory 1 */
	float x2[IOBUFFSIZE];		/* input memory 2 */
	float y[2*IOBUFFSIZE];		/* output memory */
/* Continue to 6. */
```

6. C: repeat to the end of the input files

   (a) read in one buffer of audio from each input file
   (b) interleave samples in output buffer
   (c) write out the output buffer

```
	int m, n;
	int c0 = fread(x1,sizeof(float),IOBUFFSIZE,I1);  /* read in one buffer of audio from input 1 */
	int c1 = fread(x2,sizeof(float),IOBUFFSIZE,I2);  /* read in one buffer of audio from input 2 */
	int cin = (c0<c1?c0:c1);
	while (cin>0)
	{
		for (m=0, n=0; n<cin; n++)
		{
			/* interleave samples in output buffer */
			y[m] = x1[n]; m++;	/* store input 1 to output */
			y[m] = x2[n]; m++;	/* store input 2 to output */
		}
		fwrite(y,sizeof(float),m,O); /* write out the output buffer */
		c0 = fread(x1,sizeof(float),IOBUFFSIZE,I1); /* read in next sample of audio from input 1 */
		c1 = fread(x2,sizeof(float),IOBUFFSIZE,I2); /* read in next sample of audio from input 2 */
		cin = (c0<c1?c0:c1);
	}
/* Continue to 7. */
```

7. C: close files

```
	/* close binary files */
	fclose(I1);
	fclose(I2);
	fclose(O);
	return 1;
}
```

8. MATLAB: read in the raw binary file (C output file) and write it out to output stereo audio (output media file)

```
>> bin2audio('f.bin');
Input  file = f.bin
Output file = f.wav
```

2. Convert RGB color video to grayscale video (use xylophone.mp4).

1. MATLAB: read in a color RGB video (input media file) one frame at a time and write it out to a raw binary file (C input file)

```
>> video2bin('xylophone.mp4');
Input  file = xylophone.mp4
Output file = xylophone.bin
```

2. C: determine the number of rows and columns from the raw binary file header (C input file)
3. C: write out header for grayscale video (C output file)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{
	int ndim;	/* number of dimensions; video=3 */
	int nchan;	/* number of channels; gray=1 RGB=3 */
	int d0;		/* length of first dimension; video=M M=number of rows */
	int d1;		/* length of second dimension; video=N N=number of columns */
	int d2;		/* length of third dimension; video=T T=number of frames */
} dsp_file_header;

int main(int argc, char *argv[])
{
	/* binary file read/write */
	FILE *I, *O; /* I=input O=output */
	if (NULL == (I = fopen("xylophone.bin","rb")))
	{
		printf("ERROR: Can't open xylophone.bin for input.\n");
		return 0;
	}
	if (NULL == (O = fopen("xylophone-G.bin","wb")))
	{
		printf("ERROR: Can't open xylophone-G.bin for output.\n");
		return 0;
	}

	/* header read/write */
	dsp_file_header hI, hO; /* hI=header input hO=header output */
	fread(&hI,sizeof(dsp_file_header),1,I);
	memcpy(&hO,&hI,sizeof(dsp_file_header));
	hO.nchan = 1; /* output file; gray=1 RGB=3 */
	fwrite(&hO,sizeof(dsp_file_header),1,O);
/* Continue to 4. & 5. */
```

4. C: dynamically allocate memory to hold one frame of RGB video
5. C: dynamically allocate memory to hold one frame of grayscale video

```
	/* RGB to gray */
	float *x = (float*)calloc(sizeof(float),hI.d2*hI.d0*hI.d1*hI.nchan); /* input memory */
	float *y = (float*)calloc(sizeof(float),hO.d2*hO.d0*hO.d1*hO.nchan); /* output memory */
```

6. C: repeat to the end of the raw binary file

(a) read in one frame of RGB video (C input file)
(b) convert to grayscale using the formula:

$$GRAY = (0.2989)R + (0.5870)G + (0.1140)B$$

(c) write out the grayscale frame (C output file)

```
        int t=0, m, n, p=0; /* t=frame m=row n=column p=pixel */
        float R, G, B; /* value of each color channel of a pixel */
        int fin = fread(x,sizeof(float),hI.d0*hI.d1*hI.nchan,I); /* read in one frame of RGB video */
        while (t<hI.d2)
        {
                for (m=0; m<hI.d0; m++)
                {
                        for (n=0; n<hI.d1; n++)
                        {
                                /* convert to grayscale using the formula */
                                R = 0.2989*x[((t*hI.d0+m)*hI.d1+n)*hI.nchan+0]; /* 0=red */
                                G = 0.5870*x[((t*hI.d0+m)*hI.d1+n)*hI.nchan+1]; /* 1=green */
                                B = 0.1140*x[((t*hI.d0+m)*hI.d1+n)*hI.nchan+2]; /* 2=blue */
                                y[p] = R + G + B; /* store gray value to output */
                                p++; /* advance to next pixel */
                                if (p == hI.d0*hI.d1-1)
                                {
                                        fwrite(y,sizeof(float),p,O); /* write out the grayscale frame */
                                        /* read in next frame of RGB video */
                                        fin = fread(x,sizeof(float),hI.d0*hI.d1*hI.nchan,I);
                                        p = 0; /* reset pixel count */
                                        t++; /* increment frame count */
                                }
                        }
                }

        }
/* Continue to 7. */
```

7. C: close files

```
        /* close binary files */
        fclose(I);
        fclose(O);
        return 1;
}
```

8. MATLAB: read in the raw binary file (C output file) one frame at a time and write it out to a grayscale video (output media file)

```
>> bin2video('xylophone-G.bin');
Input  file = xylophone-G.bin
Output file = xylophone-G.mp4
```