EE 257 Homework 1 Solutions by Ann Chen

Problem 1

There are 1002 bytes in each line and 272 bytes of header.

The easiest way to cut the first 272 bytes header is to use a Linux command:

dd bs=272 skip=1 if=prob1.dat of=output.dat

Problem 2

C++ code

```
#include <iostream>
#include <fstream>

using namespace std;

unsigned char * image5;
unsigned char * image8;

int main () {

ifstream infile ("prob2.dat", ios::in|ios::binary|ios::ate);
ofstream outfile ("prob2.out", ios::out | ios::binary);


if (infile.is_open())

{
  image5 = new unsigned char [500*600];
  infile.seekg (0, ios::beg);
  infile.read((char *)image5,500*600);
  infile.close();
  unsigned char tmp1;
  unsigned char tmp2;
  /*convert 5 bits to 8 bits*/
  image8 = new unsigned char [800*600];
  int indx=0;
  int n=0;
  for (int i=0;i<800*600;i++){
    if(indx<=3){
      image8[i]=(image5[n]<<indx)>>3;
      indx = indx + 5;
    }else{
      tmp1=(image5[n]<<indx)>>3;
      n=n+1;
      tmp2=image5[n]>>(11-indx);
      indx=indx-3;
      image8[i]=tmp1|tmp2;
    }
```

```
    image8[i]=(image8[i]+1)*8-1;
  }

  outfile.write ((char *)image8,800*600);
  delete[] image5;
  delete[] image8;

}

 else cout << "Unable to open file";

 return 0;

}
```

FORTRAN code:

```
program quant
  implicit none   ! this statement forces all variables to the defined

  ! reads the input binary file binary.input,
  ! Convert the 5-bits image to an 8-bits image

  !! DECLARE VARIABLES
  character*80 :: fname_in, fname_out   ! string buffers of length 80
characters
  integer :: i,n,k  ! counter variable
  integer :: io ! IO status
  integer*1, allocatable ::  array1(:),array2(:)
  integer*1 :: temp1,temp2,used

  !! READ COMMAND LINE INPUTS
  if (iargc().lt.2) then
     ! make sure the user inputs a file name, otherwise print proper usage
     print *, 'USAGE: quant input_file_name output_file_name'
     print *, '   5-bits image to 8-bits image'
     stop    ! stop if number of inputs is incorrect
  else
     call getarg(1,fname_in)      ! read input file name
     call getarg(2,fname_out)     ! read output file name
  endif

  allocate(array1(500))!a line of 5 bit input
  allocate(array2(800))!a line of 8 bit output
  open(21,file=fname_in,access='direct',recl=1*500)
  open(unit=2,file=fname_out,access='direct',&  ! & says continue to next
line
       form='unformatted',recl=1*800,status ='replace') ! open output file
  do k=1,600
    read(21,rec = k,IOSTAT=io) array1 ! reads 500 variables of 1 bytes
    n=1
    used=0
    do i=1,800
      if(used<=3)then
         temp1=array1(n)
         temp2=ibits(temp1,3-used,5)
```

```
            used=used+5
         else
            temp1=ishft(array1(n),used)
            n=n+1; !use all bits in array(n),move to the next
            temp2=ishft(array1(n),used-11) !shift unused 8-(used-3)
            used=used-3  !used bit in array(n) is 8-(11-used)
            temp1=ishft(temp1,-3)
            temp2=ior(temp1,temp2)
         endif
         array2(i)=(temp2+1)*8-1
         if(array2(i)==0)then
            array2(i)=1
         endif
      enddo
      write(2,rec=k)array2
   enddo
   close(21)
   close(2)
   deallocate(array1)
   deallocate(array2)
end program quant
```

Problem 3

The satellite velocity: $v = \sqrt{\frac{GM}{(R_{mars}+h)}} = 3.3589 \times 10^3\ m/s$

The satellite period: $T = \frac{2\pi(R_{mars}+h)}{v} = 7101.3\ s$

The data were corrected by a common reference and therefore the negative data represent "mountain" and positive values represent "valley". We can convert the data to actual height on the ground by multiply (-c/2), where c is the speed of light.

In order to map the data points to the correct ground pixel, we use the latitude and longitude grid of the earth for the calculation.

It takes satellite 7101.3 second to travel 360 degree (one cycle) in the polar orbit; therefore, the longitude spacing between each sampling point is $20 \times \frac{360}{7101.3}$ degree.

It takes Mars 88775.24 second to travel 360 degree (one cycle) in the Equatorial orbit; therefore, the latitude spacing between each sampling point is $20 \times \frac{360}{88775.24}$ degree.

Assume initially the satellite is at (lat=0,long=0) and traveling north. Therefore the next satellite location is (lat+=dlat, long+=dlong), if we keep track the satellite location, we can map all the data points into the 2D latitude-longitude grid. We need to handle the special cases when satellite passes the north/south pole or longitude 0/360 degree, as latitude and longitude are wrapped

around. Also, if multiple measurements fall into one pixel, we can simply take an average of these measurements.

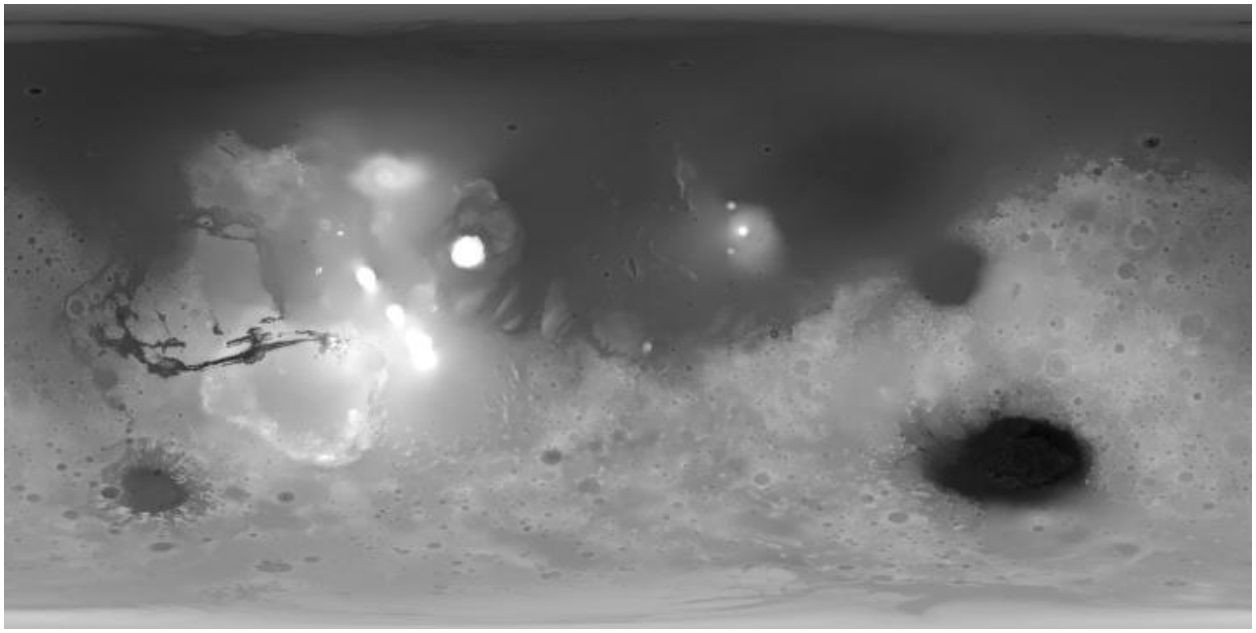

Fig.1 The Mars height map

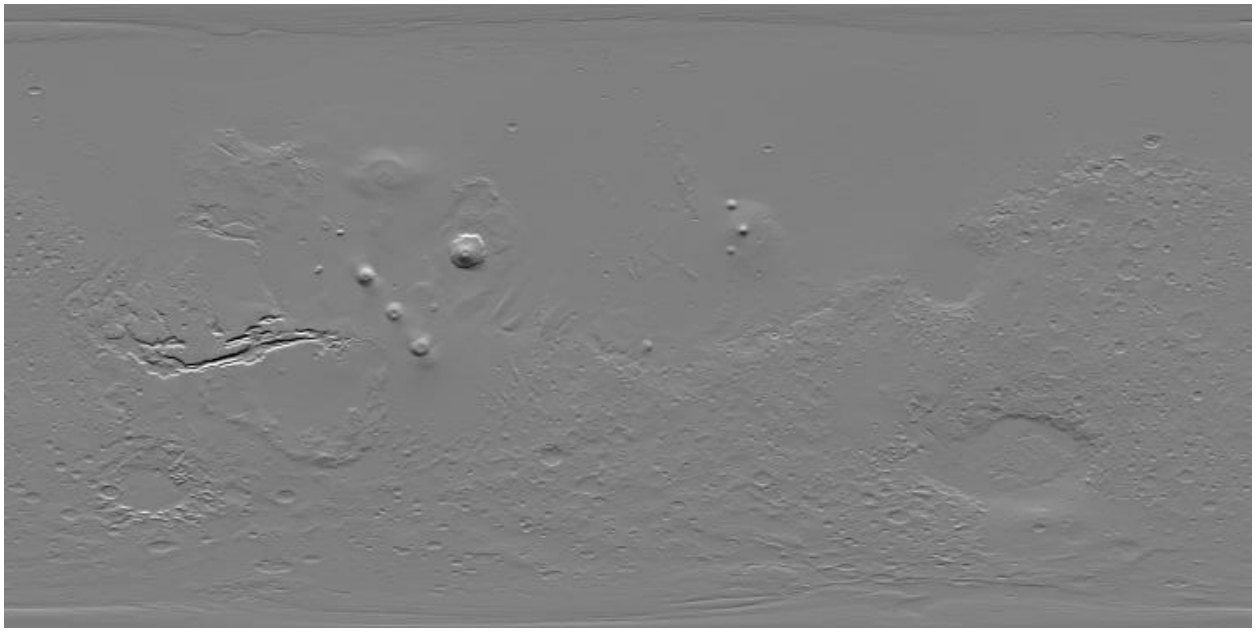

Fig.2 The Mars shaded relief image, which shows the gradient from the top to the bottom. You can also compute the shaded relief image that shows the gradient from the right to the left.

Source code for problem 3 is available; please send an email to me directly if you want a copy.