Homework 4

Brian Ackley

Cse460

1. **a**.

   P0: 0, 0, 0, 0

   P1: 0, 7, 5, 0

   P2: 1, 0, 0, 2

   P3: 0, 0, 2, 0

   P4: 0, 6, 4, 2

   b.

   yes it is in a safe state, because it still has available resources it can supply to P3, and also it can still have P0 finish which would free up more resources to use.

   c.

   yes it could be handled right away since there are resources available to take care of the added resources that are needed.

2.

```
#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "counter.txt"
using namespace std;

SDL_bool condition = SDL_FALSE;
SDL_mutex *mutex;
SDL_cond *readerQueue;
SDL_cond *writerQueue;
```

```
int readers = 0;
int writers = 0;

int reader ( void *data )
{
  SDL_LockMutex ( mutex );
  while ( !(writers == 0) )
    SDL_CondWait ( readerQueue, mutex );

  readers++;

  SDL_UnlockMutex ( mutex );

  SDL_Delay ( rand() % 3000);

  SDL_LockMutex ( mutex );
  printf(counter.txt );
  if ( --readers == 0 )
    SDL_CondSignal ( writerQueue );
  SDL_UnlockMutex ( mutex );
}

int writer ( void *data )
{
  SDL_LockMutex(mutex);
  while ( !( (readers == 0) && (writers == 0) ) )
    SDL_CondWait ( writerQueue, mutex );

  writers++;

  SDL_UnlockMutex ( mutex );
  //write
  SDL_Delay ( rand() % 3000);

  SDL_LockMutex ( mutex );
  writers--;
  printf(counter.txt);

  SDL_CondSignal ( writerQueue );
  SDL_CondBroadcast ( readerQueue );
  SDL_UnlockMutex ( mutex );
}

int main ()
{
```

```
  SDL_Thread *idr[20], *idw[3];

  mutex = SDL_CreateMutex();
  readerQueue = SDL_CreateCond();
  writerQueue = SDL_CreateCond();

  for (int i= 0; i =< counter.size; i++)
 {
    If (writer == 1)
    {
       Writers++;
       Printf(id);
    }
    If (writer == 0)
    {
       Readers++;
       Printf(id);
    }
 }
}
```

3.

```
Pthread_mutex_t mu = Pthread_mutex_intializer;
Pthread_cont_t readerQ = Pthread_cont_intializer;
Pthread_cont_t writerQ = Pthread_cont_intializer;
Int readers = 0;
Int writers = 0;
Int active_writers = 0;


Void reader ()
{
   Pthread_mutex_lock(&mu);
   While(!(writers == 0))
   {
      Pthread_cond_wait(&readerQ, &mu);
   }
   Readers++;
   Pthread_mutex_lock(&mu);
   If (--readers == 0)
      Pthread_cond_signal(&writerQ);
   Pthread_mutex_unlock(&mu);
}
```

```
Void writer()
{
   Pthread_mutex_lock(&mu);
   Writers++;
   While (!((readers == 0) && (active_writers == 0)))
      Pthread_cond_wait(&writerQ, &mu);
   Active_writers++;
   Pthread_mutex_unlock(&mu);
   Active_writers--;
   If (--writers == 0)
   {
      Pthread_cond_broadcast(&readerQ);
   }
   Else
      Pthread_cond_signal(&writerQ);
   Pthread_mutex_unlock(&mu);
}
```