#Practica6

s<-c(1, 3.5, 5, 10)

B<-c(10,20,35.33,35.5)

#si se usa sop, dentro de la función tengo que usar sop #el nombre con el que voy a llamar a polbase tiene que ser el mismo con el que voy a definir ese valor

Polbase<-function(sop,pepito,N){

L<-c(0)

for(i in 1:n){

L[i]=1

for(j in 1:N){

if(j!=i){

L[i]=L[i]\*(pepito-sop[j])/(sop[i]-sop[j])#dentro de la función tengo que usar los nombres que tengo puestos arriba

}

}

}

return(L)

}

PolInterp<-function(B, L, n){

B<-c(10,20,35.33,35.5)

p=0

for(i in 1:n){

p=p+B[i]\*L[i]

}

return(p)

}

n=length(s)

t=2

pb=Polbase(s,t,n) #fuera de la función tengo que usar los nombres definidos antes

pb

PI<-PolInterp(B, pb, n)#importante poner pb en vez de L, ya que pb es el vector que acabamos de calcular

PI

#generar 1001 abscisas equidistantes en el intervalo [s[1]-s[n]]

#usamos para ello el comando sec

x=seq(s[1], s[n], length=1001)

#otra opción-> h=(s[n]-s[1])/1000; x=seq(s[1], s[n], h) (ponemos al final el numero de intervalos o lo que mide cada trocito)

#otra opcion2-> hacer un bucle , x<-0 for(i in 1:1001){

# x[i]=s[i]+(i-1)h }

#ahora queremos obtener el valor del polinomio interpolador en esos puntos x

f=0

for(k in 1:1001){

LL<-Polbase(s,x[k],n)

f[k]<-PolInterp(B,LL,n)

}

f

plot(s,B, xlim=c(s[1],s[n]),ylim=c(0,60))

par(new='true')

#hay que poner xlim=c(s[1],s[n]),ylim=c(0,60)para que en la gráfica ambas estén en la misma escala

plot(x,f, type='l', col='blue',xlim=c(s[1],s[n]),ylim=c(0,60))