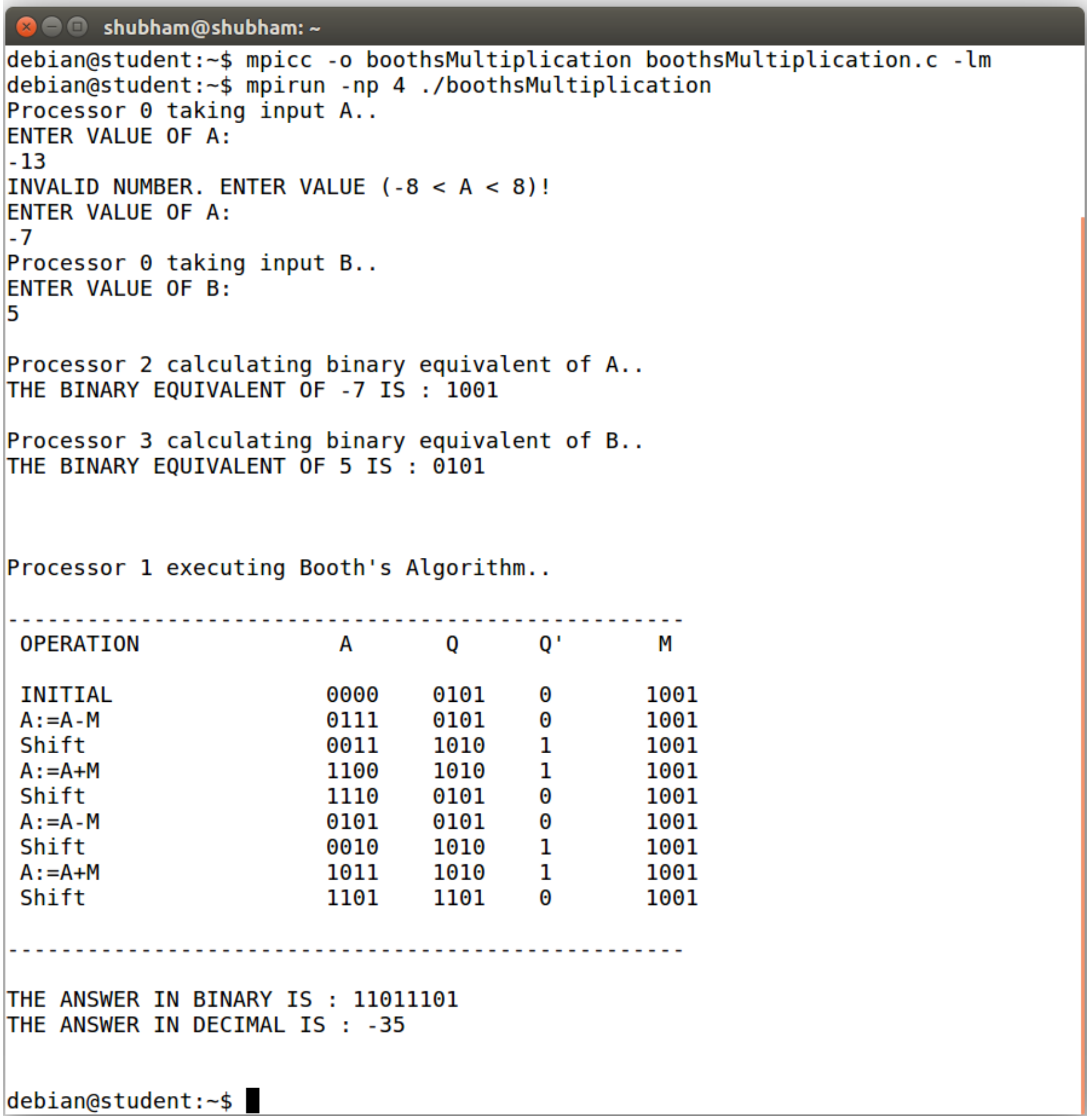
File: boothsMultiplication.c

```c
#include<stdio.h>
#include<mpi.h>
#include<math.h>

int main() {
    //initialize mpi environment
    MPI_Init(NULL,NULL);
    int rank,size;
    MPI_Status stat;
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);

    int q=0, i, j, a, b, A[4]={0,0,0,0}, C[4]={0,0,0,1},
C1[8]={0,0,0,0,0,0,0,1};
    int s=0,z=0,Q[4],M[4],temp,temp1[4],ans[8],y,flag;

    //processor 0 taking input numbers
    if(rank==0) {
        flag=0;
        printf("Processor 0 taking input A..\n");
        do {
            printf("ENTER VALUE OF A:\n");
            if(scanf("%d",&a));
            if(a<0) {
                a = a * -1;
                flag = 1;
            }
            if(8<=a)
                printf("INVALID NUMBER. ENTER VALUE (-8 < A < 8)!\n");
        }while(8<=a);
        if(flag)
            a = a*-1;

        flag=0;
        printf("Processor 0 taking input B..\n");
        do {
            printf("ENTER VALUE OF B: \n");
            if(scanf("%d",&b));
            if(b<0) {
                b = b * -1;
                flag = 1;
            }
            if(8<=b)
                printf("INVALID NUMBER. ENTER VALUE (-8 < B < 8)!\n");
        }while(8<=b);
        if(flag)
            b = b*-1;

        //send number to processor 1 and 2
        MPI_Send(&a,1,MPI_INT,2,1,MPI_COMM_WORLD);
        MPI_Send(&a,1,MPI_INT,1,1,MPI_COMM_WORLD);
        //send number to processor 1 and 3
        MPI_Send(&b,1,MPI_INT,3,2,MPI_COMM_WORLD);
        MPI_Send(&b,1,MPI_INT,1,2,MPI_COMM_WORLD);
    }

    //processor 2 calculating binary equivalent of first number
    if(rank==2) {
        MPI_Recv(&a,1,MPI_INT,0,1,MPI_COMM_WORLD,&stat);
        int i,p,c[4]={0,0,0,1};
        p=a;
        for(i=0;i< 4;i++)
            M[i] = 0;
```

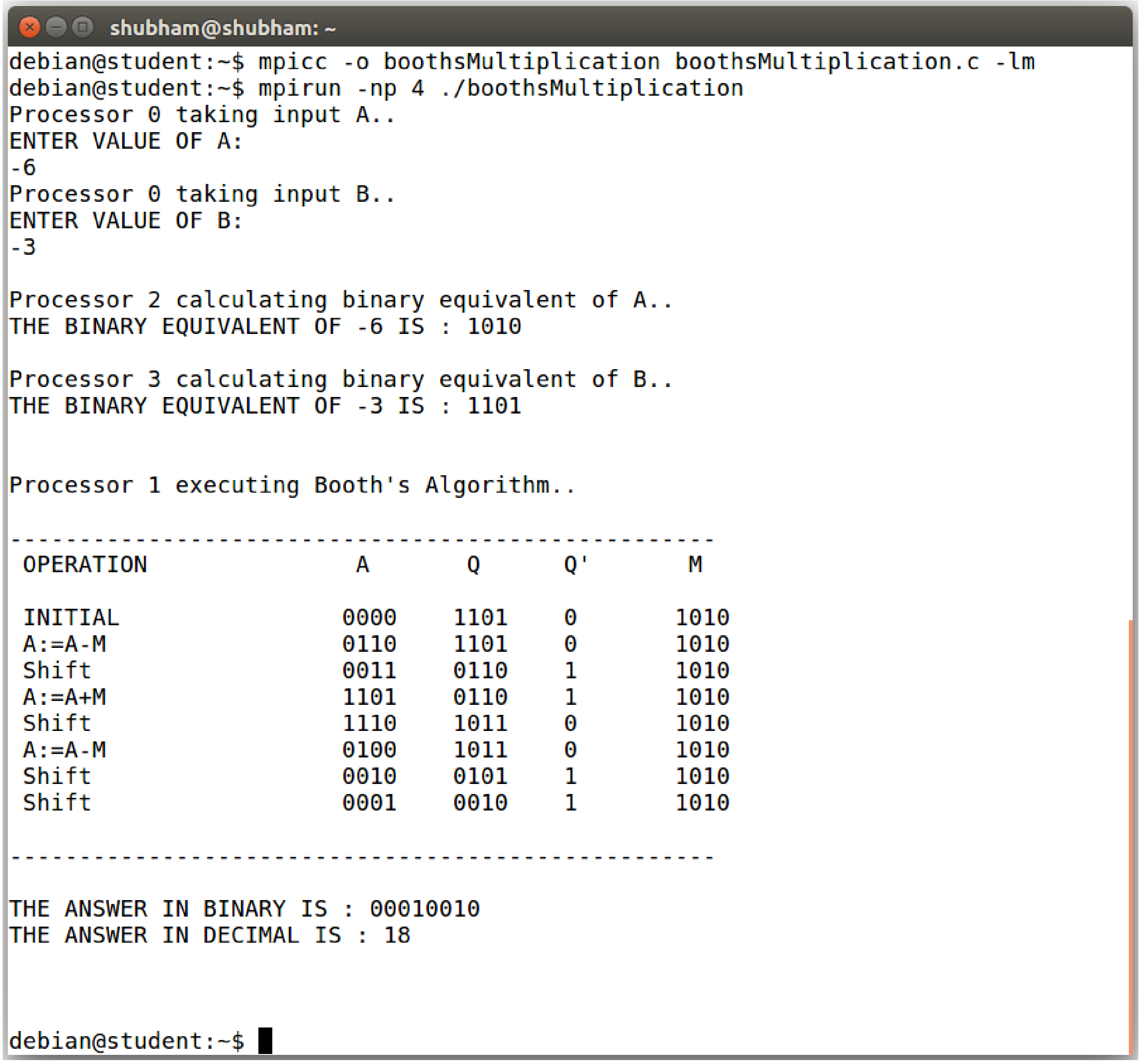
```
          //making -ve no. +ve
          if(a < 0)
               a = a *-1;
          i = 3;
          //decimal to binary conversion
          do {
               M[i]=a%2;
               a = a/2;
               i--;
          }while(a!=0);
          //2's complement
          if(p< 0) {
               //1's complement
               for(i=0;i< 4;i++)
                    M[i]=1-M[i];

               int x,i,k=0;
               //add(M,c)
               for(i=3;i>=0;i--) {
                    x=M[i];
                    M[i]=k^x^c[i];
                    if(((k==1) && (x==1)) || ((x==1) && (c[i]==1)) ||
((c[i]==1) && (k==1)))
                         k = 1;
                    else
                         k = 0;
               }
          }
          printf("\nProcessor 2 calculating binary equivalent of A..\n");
          printf("THE BINARY EQUIVALENT OF %d IS : ",p);
          for(i=0;i< 4;i++)
               printf("%d",M[i]);
          printf("\n");
     }

     MPI_Barrier(MPI_COMM_WORLD);

     //processor 3 calculating binary equivalent of second number
     if(rank==3) {
          MPI_Recv(&b,1,MPI_INT,0,2,MPI_COMM_WORLD,&stat);
          int i,p,c[4]={0,0,0,1};
          p=b;
          for(i=0;i< 4;i++)
               Q[i] = 0;
          //making -ve no. +ve
          if(b < 0)
               b = b *-1;
          i = 3;
          //decimal to binary conversion
          do {
               Q[i]=b%2;
               b = b/2;
               i--;
          }while(b!=0);
          //2's complement
          if(p< 0) {
               //1's complement
               for(i=0;i< 4;i++)
                    Q[i]=1-Q[i];
               //add(M,c);
               int x,i,k=0;
               for(i=3;i>=0;i--) {
                    x=Q[i];
                    Q[i]=k^x^c[i];
                    if(((k==1) && (x==1)) || ((x==1) && (c[i]==1)) ||
((c[i]==1) && (k==1)))
                         k = 1;
                    else
                         k = 0;
               }
          }
          printf("\nProcessor 3 calculating binary equivalent of B..\n");
          printf("THE BINARY EQUIVALENT OF %d IS : ",p);
          for(i=0;i< 4;i++)
               printf("%d",Q[i]);
          printf("\n");
     }
     //wait till all binary equivalents are calculated
     MPI_Barrier(MPI_COMM_WORLD);

     //send binary equivalents to processor 1
     if(rank==2) {
          for(i=0;i< 4;i++)
               MPI_Send(&M[i],1,MPI_INT,1,1,MPI_COMM_WORLD);
     }

     if(rank==3) {
          for(i=0;i< 4;i++)
               MPI_Send(&Q[i],1,MPI_INT,1,2,MPI_COMM_WORLD);
     }

     MPI_Barrier(MPI_COMM_WORLD);

     //procesor 1 calculating multiplication using Booth's algorithm
     if(rank==1) {
          MPI_Recv(&a,1,MPI_INT,0,1,MPI_COMM_WORLD,&stat);
          MPI_Recv(&b,1,MPI_INT,0,2,MPI_COMM_WORLD,&stat);
          for(i=0;i< 4;i++)
               MPI_Recv(&M[i],1,MPI_INT,2,1,MPI_COMM_WORLD,&stat);
          for(i=0;i< 4;i++)
               MPI_Recv(&Q[i],1,MPI_INT,3,2,MPI_COMM_WORLD,&stat);

          //Right Shift Arithmetic operation
          void rshift(int xy,int *yx) {
               int i;
               for(i=3;i>0;i--)
               yx[i] = yx[i-1];
               yx[0] = xy;
          }

          //Binary Addition
          void add(int *ab,int *ba) {
               int x,i,k=0;
               for(i=3;i>=0;i--) {
                    x=ab[i];
                    ab[i]=k^x^ba[i];
                    if(((k==1) && (x==1)) || ((x==1) && (ba[i]==1)) ||
((ba[i]==1) && (k==1)))
                         k = 1;
                    else
                         k = 0;
               }
          }
          printf("\nProcessor 1 executing Booth's Algorithm..\n");
          printf("\n----------------------------------------------------\n");
          printf(" OPERATION\t\t A\t Q\tQ'\t M\n");
          printf("\n INITIAL\t\t");
```

```
for(i=0;i< 4;i++)
                printf("%d",A[i]);
           printf("\t");
           for(i=0;i< 4;i++)
                printf("%d",Q[i]);
           printf("\t");
           printf("%d\t",q);
           for(i=0;i< 4;i++)
                printf("%d",M[i]);
           for(j=0;j< 4;j++) {
           if((Q[3]==0)&&(q==1)) {
                printf("\n A:=A+M \t\t");
                add(A,M);
                for(i=0;i< 4;i++)
                     printf("%d",A[i]);
                printf("\t");
                for(i=0;i< 4;i++)
                     printf("%d",Q[i]);
                printf("\t%d\t",q);
                for(i=0;i< 4;i++)
                     printf("%d",M[i]);
           }
           if((Q[3]==1)&&(q==0))
           {
                printf("\n A:=A-M \t\t");
                for(i=0;i< 4;i++)
                     temp1[i] = 1-M[i];
                add(temp1,C);
                add(A,temp1);
                for(i=0;i< 4;i++)
                     printf("%d",A[i]);
                printf("\t");
                for(i=0;i< 4;i++)
                     printf("%d",Q[i]);
                printf("\t%d\t",q);
                for(i=0;i< 4;i++)
                     printf("%d",M[i]);
           }
           printf("\n Shift \t\t\t");
           y = A[3];
           q = Q[3];
           rshift(A[0],A);
           rshift(y,Q);
           for(i=0;i< 4;i++)
                printf("%d",A[i]);
           printf("\t");
           for(i=0;i< 4;i++)
                printf("%d",Q[i]);
           printf("\t");
           printf("%d\t",q);
           for(i=0;i< 4;i++)
                printf("%d",M[i]);
           }
           //calculate answer in binary
           printf("\n\n--------------------------------------------------\n");
           printf("\nTHE ANSWER IN BINARY IS : ");
           for(i=0;i< 4;i++)
                ans[i]=A[i];
           for(i=0;i< 4;i++)
                ans[i+4]=Q[i];
           for(i=0;i< 8;i++)
                printf("%d",ans[i]);
           if(((a< 0)&&(b>0))||((a>0)&&(b< 0))) {
            for(i=0;i< 8;i++)
                ans[i]=1-ans[i];
            int k=0;
            for(i=7;i>=0;i--)
            {
                int x = ans[i];
                ans[i]=k^x^C1[i];
                if(((k==1) && (x==1)) || ((x==1) && (C1[i]==1)) ||
((C1[i]==1) && (k==1)))
                k=1;
                else
                k=0;
            }
        }
        //calculate answer in decimal
        for(i=7;i>=0;i--) {
            s = s + (pow(2,z) * ans[i]);
            z = z+1;
        }
        if(((a< 0)&&(b>0))||((a>0)&&(b< 0)))
            printf("\nTHE ANSWER IN DECIMAL IS : -%d\n",s);
        else
            printf("\nTHE ANSWER IN DECIMAL IS : %d\n",s);
    }
    printf("\n");
    MPI_Finalize();
    return 0;
}
```

#OUTPUT:



```
debian@student:~$ mpicc -o boothsMultiplication boothsMultiplication.c -lm
debian@student:~$ mpirun -np 4 ./boothsMultiplication
Processor 0 taking input A..
ENTER VALUE OF A:
-13
INVALID NUMBER. ENTER VALUE (-8 < A < 8)!
ENTER VALUE OF A:
-7
Processor 0 taking input B..
ENTER VALUE OF B:
5

Processor 2 calculating binary equivalent of A..
THE BINARY EQUIVALENT OF -7 IS : 1001

Processor 3 calculating binary equivalent of B..
THE BINARY EQUIVALENT OF 5 IS : 0101



Processor 1 executing Booth's Algorithm..

---------------------------------------------------
 OPERATION               A       Q      Q'       M

 INITIAL                0000    0101    0       1001
 A:=A-M                 0111    0101    0       1001
 Shift                  0011    1010    1       1001
 A:=A+M                 1100    1010    1       1001
 Shift                  1110    0101    0       1001
 A:=A-M                 0101    0101    0       1001
 Shift                  0010    1010    1       1001
 A:=A+M                 1011    1010    1       1001
 Shift                  1101    1101    0       1001

---------------------------------------------------

THE ANSWER IN BINARY IS : 11011101
THE ANSWER IN DECIMAL IS : -35


debian@student:~$ 
```



```
debian@student:~$ mpicc -o boothsMultiplication boothsMultiplication.c -lm
debian@student:~$ mpirun -np 4 ./boothsMultiplication
Processor 0 taking input A..
ENTER VALUE OF A:
-6
Processor 0 taking input B..
ENTER VALUE OF B:
-3

Processor 2 calculating binary equivalent of A..
THE BINARY EQUIVALENT OF -6 IS : 1010

Processor 3 calculating binary equivalent of B..
THE BINARY EQUIVALENT OF -3 IS : 1101


Processor 1 executing Booth's Algorithm..

---------------------------------------------------
 OPERATION                A       Q       Q'       M

 INITIAL                 0000    1101    0       1010
 A:=A-M                  0110    1101    0       1010
 Shift                   0011    0110    1       1010
 A:=A+M                  1101    0110    1       1010
 Shift                   1110    1011    0       1010
 A:=A-M                  0100    1011    0       1010
 Shift                   0010    0101    1       1010
 Shift                   0001    0010    1       1010

---------------------------------------------------

THE ANSWER IN BINARY IS : 00010010
THE ANSWER IN DECIMAL IS : 18



debian@student:~$ █
```