1. Write a C program for implementing the algorithm of Binary Addition of Signed Magnitude Number.

Source Code:

```
#include <stdio.h>
int main()
{
   long binary1, binary2;
   int i = 0, remainder = 0, sum[20];

   printf("Enter the first binary number: ");
   scanf("%ld", &binary1);
   printf("Enter the second binary number: ");
   scanf("%ld", &binary2);
   while (binary1 != 0 || binary2 != 0)
   {
      sum[i++] =(binary1 % 10 + binary2 % 10 + remainder) % 2;
      remainder =(binary1 % 10 + binary2 % 10 + remainder) / 2;
      binary1 = binary1 / 10;
      binary2 = binary2 / 10;
   }
   if (remainder != 0)
      sum[i++] = remainder;
   --i;
   printf("Sum of two binary numbers: ");
   while (i >= 0)
      printf("%d", sum[i--]);
   return 0;
}
```

Output:

```
Enter the first binary number: 111
Enter the second binary number: 010
Sum of two binary numbers: 1001
--------------------------------
Process exited after 12.29 seconds with return value 0
Press any key to continue . . .
```

2. Write a C program for implementing algorithm of Binary Subtraction using 2's Complement.

Source Code:

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

void main(){

        int i,numa[8]={0},numb[8]={0},diff[8]={0};
        printf("SUBTRACTION USING TWO'S COMPLEMENT");
        printf("\nEnter two 8-bit binary numbers\n");

        printf("\nEnter first number:\n ");
for(i=0;i<8;i++)
    {
        scanf("%d",&numa[i]);
}
printf("\nEnter second number:\n ");
for(i=0;i<8;i++)
{
        scanf("%d",&numb[i]);
}

for(i=7; i>=0;i--){
        diff[i] = numa[i] - numb[i];
        if(diff[i] < 0){
                numa[i-1] = numa[i-1] - 1;
        }
        diff[i] = fabs(diff[i]%2);
}
printf("\nDifference is: ");
for(i=0;i<8;i++){
        printf("%d",diff[i]);
  }
}
```

Output:

```
SUBTRACTION USING TWO'S COMPLEMENT
Enter two 8-bit binary numbers

Enter first number:
0
0
0
0
0
1
1
1

Enter second number:
 0
0
0
0
0
1
1
1

Difference is: 00000100
--------------------------------
Process exited after 21.93 seconds with return value 0
Press any key to continue . . .
```

3. Write a C program for implementing Booth's Multiplication Algorithm.

Source Code:

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

int a = 0,b = 0, c = 0, a1 = 0, b1 = 0, com[5] = { 1, 0, 0, 0, 0};
int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};
int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};

void binary(){
    a1 = fabs(a);
    b1 = fabs(b);
  int r, r2, i, temp;
  for (i = 0; i < 5; i++){
        r = a1 % 2;
        a1 = a1 / 2;
        r2 = b1 % 2;
        b1 = b1 / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if(r2 == 0){
            bcomp[i] = 1;
        }
        if(r == 0){
            acomp[i] =1;
        }
  }
//part for two's complementing
c = 0;
for ( i = 0; i < 5; i++){
        res[i] = com[i]+ bcomp[i] + c;
        if(res[i] >= 2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i] % 2;
  }
for (i = 4; i >= 0; i--){
  bcomp[i] = res[i];
```

```
}
//in case of negative inputs
if (a < 0){
  c = 0;
 for (i = 4; i >= 0; i--){
      res[i] = 0;
 }
 for ( i = 0; i < 5; i++){
      res[i] = com[i] + acomp[i] + c;
      if (res[i] >= 2){
          c = 1;
      }
      else
          c = 0;
      res[i] = res[i]%2;
   }
   for (i = 4; i >= 0; i--){
      anum[i] = res[i];
      anumcp[i] = res[i];
   }

  }
  if(b < 0){
   for (i = 0; i < 5; i++){
      temp = bnum[i];
      bnum[i] = bcomp[i];
      bcomp[i] = temp;
   }
  }
}
void add(int num[]){
  int i;
c = 0;
for ( i = 0; i < 5; i++){
      res[i] = pro[i] + num[i] + c;
      if (res[i] >= 2){
          c = 1;
      }
      else{
          c = 0;
      }
      res[i] = res[i]%2;
 }
 for (i = 4; i >= 0; i--){
```

```
        pro[i] = res[i];
        printf("%d",pro[i]);
     }
   printf(":");
   for (i = 4; i >= 0; i--){
         printf("%d", anumcp[i]);
     }
}
void arshift(){//for arithmetic shift right
    int temp = pro[4], temp2 = pro[0], i;
    for (i = 1; i < 5  ; i++){//shift the MSB of product
      pro[i-1] = pro[i];
    }
    pro[4] = temp;
    for (i = 1; i < 5  ; i++){//shift the LSB of product
       anumcp[i-1] = anumcp[i];
    }
    anumcp[4] = temp2;
    printf("\nAR-SHIFT: ");//display together
    for (i = 4; i >= 0; i--){
       printf("%d",pro[i]);
    }
    printf(":");
    for(i = 4; i >= 0; i--){
       printf("%d", anumcp[i]);
    }
}

void main(){
   int i, q = 0;
   printf("\t\tBOOTH'S MULTIPLICATION ALGORITHM");
   printf("\nEnter two numbers to multiply: ");
   printf("\nBoth must be less than 16");
//simulating for two numbers each below 16
do{
     printf("\nEnter A: ");
     scanf("%d",&a);
     printf("Enter B: ");
     scanf("%d", &b);
  }while(a >=16 || b >=16);

 printf("\nExpected product = %d", a * b);
 binary();
 printf("\n\nBinary Equivalents are: ");
```

```
printf("\nA = ");
for (i = 4; i >= 0; i--){
   printf("%d", anum[i]);
}
printf("\nB = ");
for (i = 4; i >= 0; i--){
   printf("%d", bnum[i]);
}
printf("\nB'+ 1 = ");
for (i = 4; i >= 0; i--){
   printf("%d", bcomp[i]);
}
printf("\n\n");
for (i = 0;i < 5; i++){
      if (anum[i] == q){//just shift for 00 or 11
         printf("\n-->");
         arshift();
         q = anum[i];
      }
      else if(anum[i] == 1 && q == 0){//subtract and shift for 10
        printf("\n-->");
        printf("\nSUB B: ");
        add(bcomp);//add two's complement to implement subtraction
        arshift();
        q = anum[i];
      }
      else{//add ans shift for 01
        printf("\n-->");
        printf("\nADD B: ");
        add(bnum);
        arshift();
        q = anum[i];
      }
   }

   printf("\nProduct is = ");
   for (i = 4; i >= 0; i--){
      printf("%d", pro[i]);
   }
   for (i = 4; i >= 0; i--){
      printf("%d", anumcp[i]);
   }
}
```

Output:

```
                BOOTH'S MULTIPLICATION ALGORITHM
Enter two numbers to multiply:
Both must be less than 16
Enter A: 12
Enter B: 15

Expected product = 180

Binary Equivalents are:
A = 01100
B = 01111
B'+ 1 = 10001



-->
AR-SHIFT: 00000:00110
-->
AR-SHIFT: 00000:00011
-->
SUB B: 10001:00011
AR-SHIFT: 11000:10001
-->
AR-SHIFT: 11100:01000
-->
ADD B: 01011:01000
AR-SHIFT: 00101:10100
Product is = 0010110100
--------------------------------
Process exited after 11.69 seconds with return value 1
Press any key to continue . . .
```

4. Write a C program for implementing Restoring Division Algorithm.

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int a=0,b=0,c=0,com[5]={1,0,0,0,0},s=0;
int anum[5]={0},anumcp[5] ={0},bnum[5]={0};
int acomp[5]={0},bcomp[5]={0},rem[5]={0},quo[5]={0},res[5]={0};

void binary(){
    a = fabs(a);
    b = fabs(b);
```

```
int r, r2, i, temp;
for(i = 0; i < 5; i++){
    r = a % 2;
    a = a / 2;
    r2 = b % 2;
    b = b / 2;
    anum[i] = r;
    anumcp[i] = r;
    bnum[i] = r2;
    if(r2 == 0){
        bcomp[i] = 1;
    }
    if(r == 0){
        acomp[i] =1;
    }
  }
  //part for two's complementing
  c = 0;
  for( i = 0; i < 5; i++){
        res[i] = com[i]+ bcomp[i] + c;
        if(res[i]>=2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i]%2;
    }
  for(i = 4; i>= 0; i--){
    bcomp[i] = res[i];
  }
}
void add(int num[]){
    int i;
    c = 0;
    for( i = 0; i < 5; i++){
        res[i] = rem[i]+ num[i] + c;
        if(res[i]>=2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i]%2;
    }
    for(i = 4; i>= 0; i--){
```

```
        rem[i] = res[i];
        printf("%d",rem[i]);
    }
    printf(":");
    for(i = 4; i>= 0; i--){
        printf("%d",anumcp[i]);
    }
}
void shl(){//for shift left
    int i;
    for(i = 4; i > 0  ; i--){//shift the remainder
        rem[i] = rem[i-1];
    }
    rem[0] = anumcp[4];
    for(i = 4; i > 0  ; i--){//shift the remtient
        anumcp[i] = anumcp[i-1];
    }
    anumcp[0] = 0;
    printf("\nSHIFT LEFT: ");//display together
    for(i = 4; i>= 0; i--){
        printf("%d",rem[i]);
    }
    printf(":");
    for(i = 4; i>= 0; i--){
        printf("%d",anumcp[i]);
    }
}

void main(){

    int i;
    printf("\t\tRESTORING DIVISION ALGORITHM");
    printf("\nEnter two numbers to Divide: ");
    printf("\nBoth must be less than 16");
    //simulating for two numbers each below 16
    do{
        printf("\nEnter A: ");
        scanf("%d",&a);
        printf("Enter B: ");
        scanf("%d",&b);
    }while(a>=16 || b>=16);

    printf("\nExpected Quotient = %d", a/b);
    printf("\nExpected Remainder = %d", a%b);
```

```
if(a*b <0){
     s = 1;
}

binary();
printf("\n\nUnsigned Binary Equivalents are: ");
printf("\nA = ");
for(i = 4; i>= 0; i--){
     printf("%d",anum[i]);
}
printf("\nB = ");
for(i = 4; i>= 0; i--){
     printf("%d",bnum[i]);
}
printf("\nB'+ 1 = ");
for(i = 4; i>= 0; i--){
     printf("%d",bcomp[i]);
}
printf("\n\n-->");
//division part
shl();
for(i=0;i<5;i++){
     printf("\n-->"); //start with subtraction
     printf("\nSUB B: ");
     add(bcomp);
     if(rem[4]==1){//simply add for restoring
          printf("\n-->RESTORE");
          printf("\nADD B: ");
          anumcp[0] = 0;
          add(bnum);
     }
     else{
          anumcp[0] = 1;
     }
     if(i<4)
          shl();

}
printf("\n---------------------------");
printf("\nSign of the result = %d",s);
printf("\nRemainder is = ");
for(i = 4; i>= 0; i--){
     printf("%d",rem[i]);
}
```

```
    printf("\nQuotient is = ");
    for(i = 4; i>= 0; i--){
        printf("%d",anumcp[i]);
    }
getch();
}
```

Output:

```
                RESTORING DIVISION ALGORITHM
Enter two numbers to Divide:
Both must be less than 16
Enter A:  10
Enter B:  9

Expected Quotient = 1
Expected Remainder = 1

Unsigned Binary Equivalents are:
A = 01010
B = 01001
B'+ 1 = 10111
-->
SHIFT LEFT: 00000:10100
-->
SUB B: 10111:10100
-->RESTORE
ADD B: 00000:10100
SHIFT LEFT: 00001:01000
-->
SUB B: 11000:01000
-->RESTORE
ADD B: 00001:01000
SHIFT LEFT: 00010:10000
-->
SUB B: 11001:10000
-->RESTORE
ADD B: 00010:10000
SHIFT LEFT: 00101:00000
-->
SUB B: 11100:00000
-->RESTORE
ADD B: 00101:00000
SHIFT LEFT: 01010:00000
-->
SUB B: 00001:00000
----------------------------
Sign of the result = 0
Remainder is = 00001
```

```
Quotient is = 00001
```