Punyaja Mishra

00660001

COIS 3320H : Final Exam

Q1. Who is your favorite lecturer of all time? (2 points)

Well this is a tough question. I have like a few favorite lectures. Until now I had Richard Hurley and Jamie Mitchell. Now, not to flatter anyone, but Alaadin A is also added on the list. These all professors keep the class engrossed and at least I don't feel bored or lost.

Q2. Page Reference Strings (4 points)

In a four-frame system where all frames are initially empty. how many page faults would occur for the optimal, FIFO, and LRU algorithm; given the following page reference string:

1 2 1 3 4 5 1 0 5 4 6 3 2 1 3 2 0 1 4 3

Make sure you show all your work. A table would be optimal to show your work for this question!

Optimal : 10-page faults

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RS		1	2	1	3	4	5	1	0	5	4	6	3	2	1	3	2	0	1	4	3
Frame0		1	1	1	1	1	1	~~1~~	0	0	0	0	0	0	0	0	0	0	0	0	0
Frame1			2	2	2	~~2~~	5	5	5	5	~~5~~	6	~~6~~	2	2	2	2	2	~~2~~	4	4
Frame2					3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Frame3						4	4	4	4	4	4	4	4	~~4~~	1	1	1	1	1	1	1
Page Fault		*	*		*	*	*		*			*		*	*					*	

FIFO : 14 - page faults

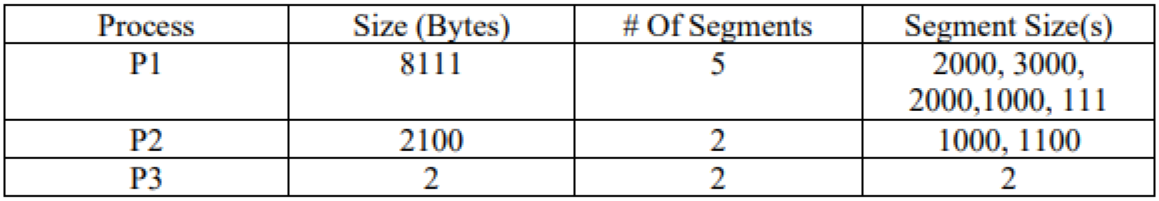
Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RS		1	2	1	3	4	5	1	0	5	4	6	3	2	1	3	2	0	1	4	3
Frame 0		> 1	> 1	> 1	> 1	> ~~1~~	5	5	5	5	5	> ~~5~~	3	3	3	3	3	> 3	> ~~3~~	4	4
Frame 1			2	2	2	2	> ~~2~~	1	1	1	1	1	> ~~1~~	2	2	2	2	2	2	> ~~2~~	3
Frame 2					3	3	3	> ~~3~~	0	0	0	0	0	> ~~0~~	1	1	1	1	1	1	1
Frame 3						4	4	4	> 4	> 4	> ~~4~~	6	6	6	> 6	> 6	> ~~6~~	0	0	0	0
Page Fault		*	*		*	*	*	*	*			*	*	*	*			*		*	*

LRU : 13 – page faults

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RS		1	2	1	3	4	5	1	0	5	4	6	3	2	1	3	2	0	1	4	3
Frame0		1	1	1	1	1	1	1	1	1	~~1~~	6	6	6	6	6	~~6~~	0	0	0	0
Frame1			2	2	2	~~2~~	5	5	5	5	5	5	~~5~~	2	2	2	2	2	2	~~2~~	3
Frame2					3	3	3	~~3~~	0	0	0	~~0~~	3	3	3	3	3	3	~~3~~	4	4
Frame3						4	4	4	4	4	4	4	4	~~4~~	1	1	1	1	1	1	1
Page Fault		*	*		*	*	*		*			*	*	*	*			*		*	*

Q3. Paging and Segmentation (8 points in total, 2 points each)

Consider the following list of processes and their associated sizes.



Process	Size (Bytes)	# Of Segments	Segment Size(s)
P1	8111	5	2000, 3000, 2000,1000, 111
P2	2100	2	1000, 1100
P3	2	2	2

a. **Assuming that Paging is used, show the page table for each process under the following scenario: assume a page size of 2000 bytes and a main memory of size 32 frames with Frames 0, 2, 4, 5, 7, 10, 15, 16, 17, 21, 24 and 31 currently utilized. Also assume the free frames are ordered by number in the free-space list. Each Page Table entry should contain Page# and Frame#**

Page size = 2000

Main memory size = 32 frames

Utilized frames :

0, 2, 4, 5, 7, 10, 15, 16, 17, 21, 24, 31

P1

8111 bytes = 8111/2000 = 4.05 = 5 pages

P1	
Page #	Frame #
0	1
1	3
2	6
3	8
4	9

Updated utilized frames : 0 1 2 3 4 5 6 7 8 9 10 15 16 17 21 24 31

P2

2100 bytes = 2100/2000 = 1.05 = 2 pages

P2	
Page #	Frame #
0	11
1	12

Updated utilized frames : 0 1 2 3 4 5 6 7 8 9 10 11 12 15 16 17 21 24 31

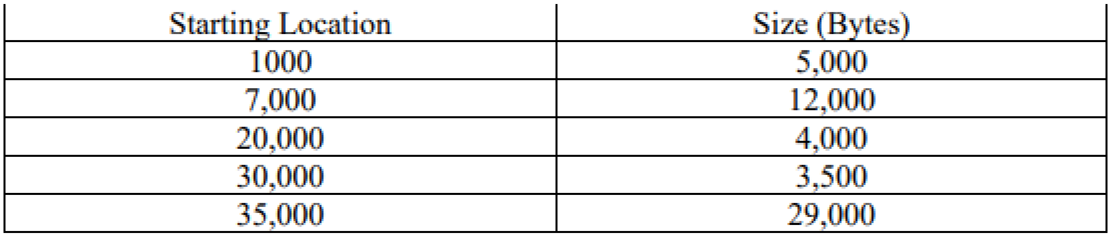
P3

8111 bytes = 2/2000 = 0.001 = 1 pages

P3	
Page #	Frame #
0	13

Updated utilized frames : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 21 24 31

b. **Assuming that Segmentation is used, show the Segment Table for each process using the next fit allocation algorithm under the following scenario: given a main memory size of 64,000 bytes with the following areas currently available. Assume a segment is added at the beginning of the free hole. Each segment table should contain the segment #, size, and the MM address.**



Starting Location	Size (Bytes)
1000	5,000
7,000	12,000
20,000	4,000
30,000	3,500
35,000	29,000

Next – fit : The next location is chosen from the last allocation location

P1

5 segments

Segment Sizes : 2000, 3000, 2000, 1000, 111

P1		
Segment #	Segment Size	Address
0	2000	1000 – 3000
1	3000	7000 - 10000
2	2000	20000 – 22000
3	1000	30000 – 31000
4	111	35000 - 36111

P2

2 segments

Segment Sizes : 1000, 1100

P2		
Segment #	Segment Size	Address
0	1000	3000 – 4000
1	1100	10000 - 11000

P3

1 segments

Segment Sizes : 2

P1		
Segment #	Segment Size	Address
0	2	22000 – 22002

c. **Repeat part b assuming worst fit allocation is used.**

Worst Fit – The worst available hole (that leaves a lot of space is chosen and the entire thing is traversed)

P1

5 segments

Segment Sizes : 2000, 3000, 2000, 1000, 111

P1		
Segment #	Segment Size	Address
0	2000	35000 – 37000
1	3000	37000 – 40000
2	2000	40000 – 42000
3	1000	42000 – 43000
4	111	43000 – 43111

P2

2 segments

Segment Sizes : 1000, 1100

P2		
Segment #	Segment Size	Address
0	1000	43111 – 44111
1	1100	44111 - 45211

P3

1 segments

Segment Sizes : 2

P1		
Segment #	Segment Size	Address
0	2	45211 - 45213

d. **Assuming that Paged-Segmentation is used, show the Segment and Pages Tables for each process using the same scenario for main memory as described in Part (a). The Segment Tables entries for this part should contain Segment#, Size, and Page Table id (start counting at 0).**

V

Segmentation with Paging

Page Size – 2000 bytes

Main memory – 32 frames size

Segment Size – differs for every segment

The memory location for pages are used as given in part a question.

Currently utilized frames :

0, 2, 4, 5, 7, 10, 15, 16, 17, 21, 24, 31

P1 needs 5 pages

P2 needs 2 pages

P3 needs 1 page

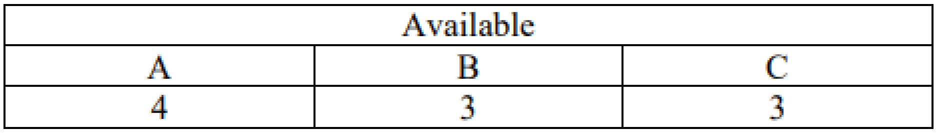
Process	Page Table Memory Id	Number of Pages	Size (used)
P1	1, 3, 6, 8, 9	5	4*2000 + 1*111
P2	11, 12	2	1*2000 + 1*100
P3	13	1	1*2

Process	Page Table ID	Page Table Memory Id	Segment #	Size (used)
P1	0	1, 3, 6, 8, 9	0	8111
P2	1	11, 12	1	2100
P3	2	13	2	2

Q4. Banker's Algorithm (6 points 2 points each)

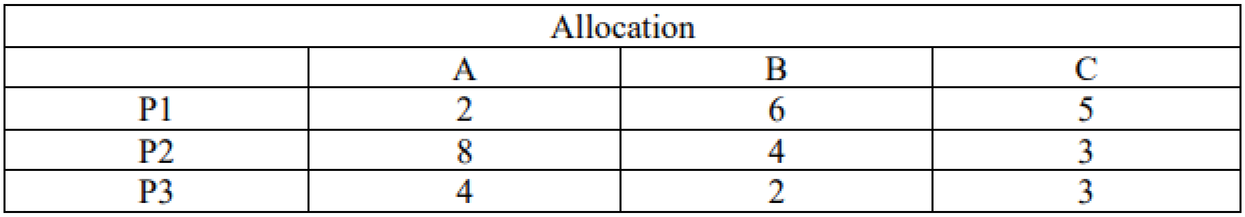
Given the following snapshot of a system:

Available :



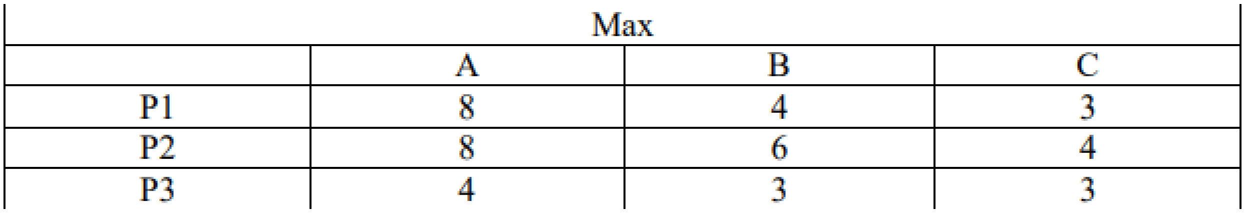
Available		
A	B	C
4	3	3

Allocation Matrix :



Allocation			
	A	B	C
P1	2	6	5
P2	8	4	3
P3	4	2	3

Max Matrix :



Max			
	A	B	C
P1	8	4	3
P2	8	6	4
P3	4	3	3

a. **Is there an error in the max matrix? If yes, correct it**
The max matrix sets the limit to the maximum amount of resources each process can demand in the system. This means the allocated number of resources can not go over the maximum number of resources.

For P1 Resource - B, Max matrix says 4 but allocated resources are 6. This means that max matrix should probably say 6.
For P1 Resource – C, Max Matrix says 3 but allocated resources are 5. This means that max matrix should say 5.

Max			
	A	B	C
P1	8	6	5
P2	8	6	4
P3	4	3	3

b. **- Is the system in a safe state (is there a sequence of processes which will not lead to a deadlock)? Show the safe sequence if yes, or show why it cannot run due to a deadlock if the answer is no.**
Need Matrix = Max Matrix – Allocation Matrix

Need			
	A	B	C
P1	8 – 2 = 6	6 – 6 = 0	5 – 5 = 0
P2	8 – 8 = 0	6 – 4 = 2	4 – 3 = 1
P3	4 – 4 = 0	3 – 2 = 1	3 – 3 = 0

Available :

A	B	C
4	3	3

A process can be run if need < = available

P1 can not run (Resource A 6 > 4)

P2 – 0 < 4; 2 < 3; 1 < 3 Therefore P2 can run.

The P2 releases its resources. Allocation to P2 was 8, 4, 3; so now the available matrix is

A	B	C
4 + 8 = 12	3 + 4 = 7	3 + 3 = 6

Now P1 can run. P1 releases its resources. Allocation to P1 was 2, 6, 5; so now the available matrix is

A	B	C
12 + 2 = 14	7 + 6 = 13	6 + 5 = 11

Now P3 can run.

Therefore, the system is in safe state. And the safe sequence is :
P2 -> P1 -> P3

c. **Can a request P3(5,3,2) be granted?**

After this request is granted, the new allocation matrix, new need matrix and new available matrix would be :

New Allocation matrix :

Allocation			
	A	B	C
P1	2	6	5
P2	8	4	3
P3	9	5	5

New need matrix :

Need			
	A	B	C
P1	8 – 2 = 6	6 – 6 = 0	5 – 5 = 0
P2	8 – 8 = 0	6 – 4 = 2	4 – 3 = 1
P3	-5 = 0	-2 = 0	-2 = 0

Available matrix :

A	B	C
-1	0	1

Thus the request ca not be granted because the available resources go to negative that is not possible.

Q5. **Page Fault Frequency Algorithm (4 points)**

Physical memory is initially empty. The following reference string is processed:
0 1 4 0 2 0 0 1 0 3 0 4 0 3
Show which pages are resident under the page fault frequency algorithm with d = 3.
Indicate when page faults occur.
Using a table to show your work might be a good idea for this one!

d = 3
According to the working-set page replacement algorithm

Time t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
RS		0	1	4	0	2	0	0	1	0	3	0	4	0	3
Page 0		X	X	X	X	X	X	X	X	X	X	X	X	X	X
Page 1		-	X	X	X	X	X	X	X	X	X	X	X	X	X
Page 2		-	-	-	-	X	X	X	X	X	X	X	X	X	X
Page 3		-	-	-	-	-	-	-	-	-	X	X	X	X	X
Page 4		-	-	X	X	X	X	X	X	X	-	-	X	X	X
Page Fault		*	*	*		*					*		*		
Size		1	2	3	3	4	4	4	4	4	5				

Since the distance between current and last page fault at time t = 10 is greater than d (10-5 = 5 > 3), page 4 which has not been referenced since time 5 is removed from resident set.

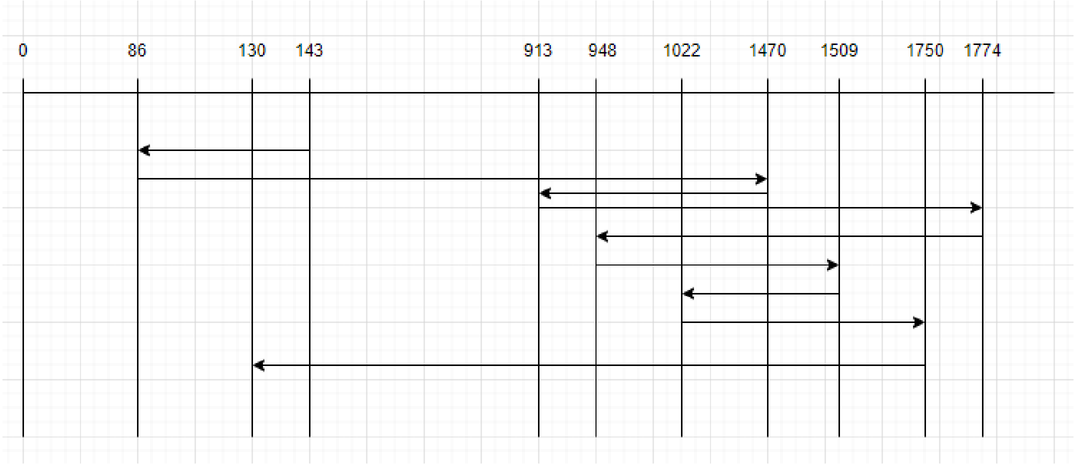
Q6. **Disk Scheduling Algorithms (8 points, 2 points each)**

The r/w head of a disk is at track 143. The previous position was track 0. Requests to access the following tracks have arrived:

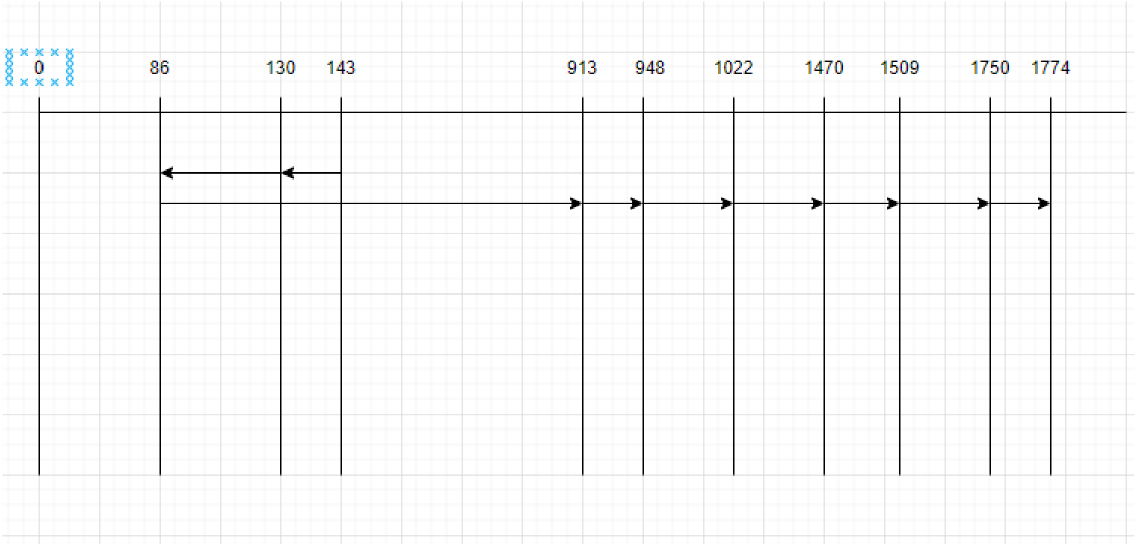
143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

In which order will the tracks be visited when using :

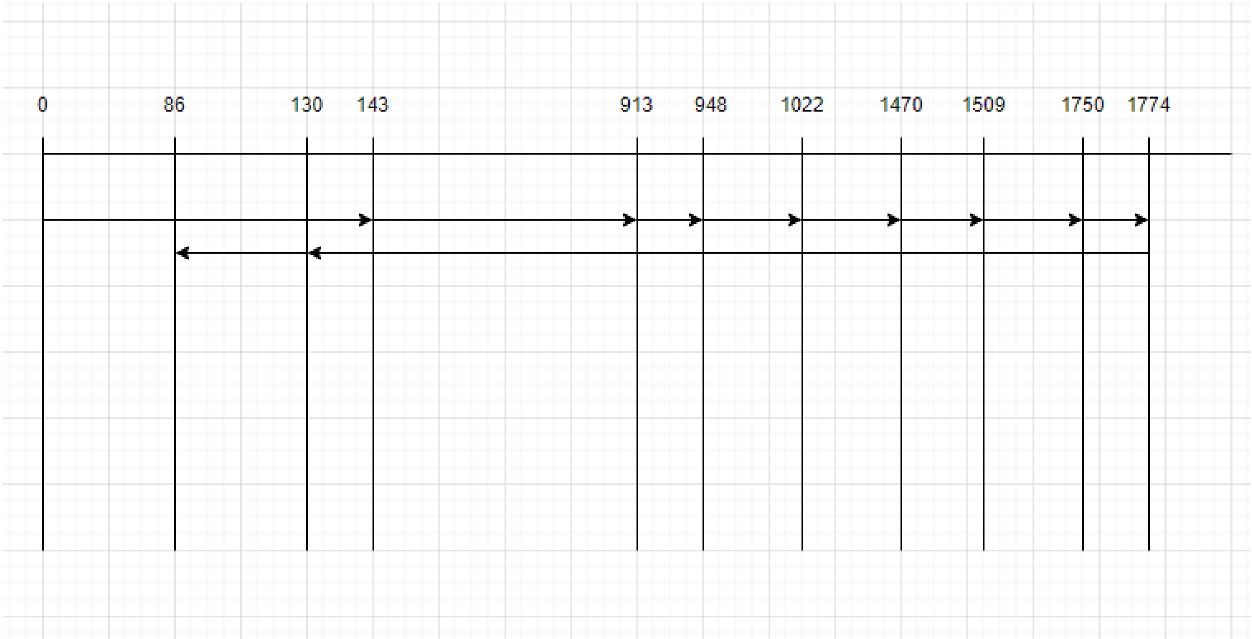
a. FIFO



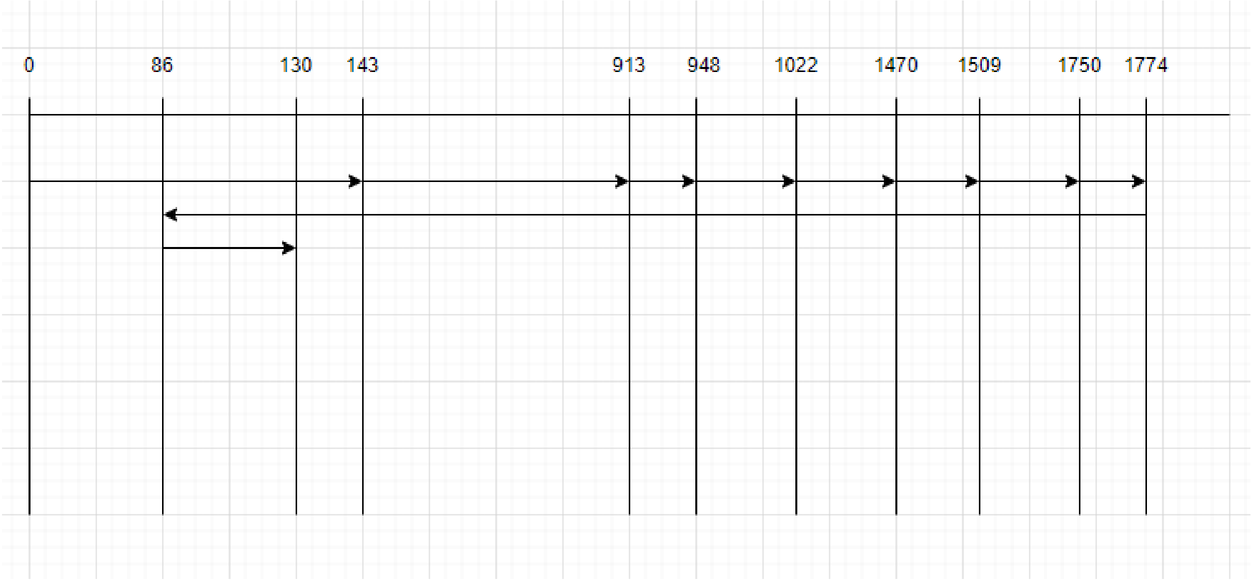
b. SSTF



c. **Scan**

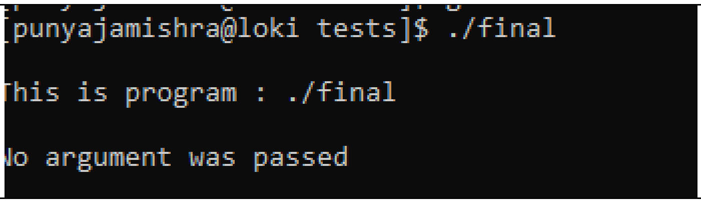


d. **C- Scan**



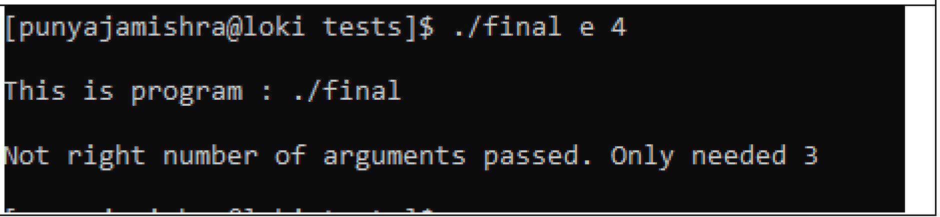
Q7. C program 10 points

Test 1	
Input	No arguments passed
Expected Output	Prints error

Actual Output	
	`[punyajamishra@loki tests]$ ./final`
	`This is program : ./final`
	`No argument was passed`

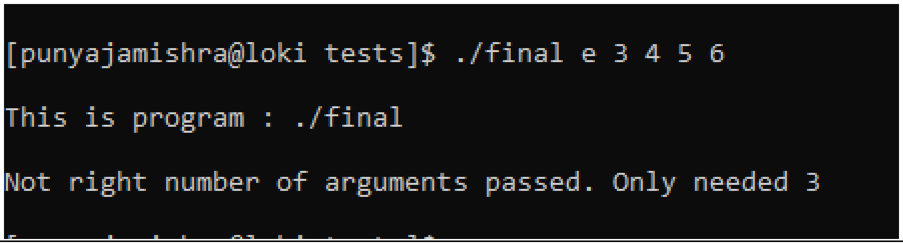
Only the file is called and nothing else is passed

Test 2	
Input	Less than expected arguments passed
Expected Output	Prints error

Actual Output	
	`[punyajamishra@loki tests]$ ./final e 4`
	`This is program : ./final`
	`Not right number of arguments passed. Only needed 3`

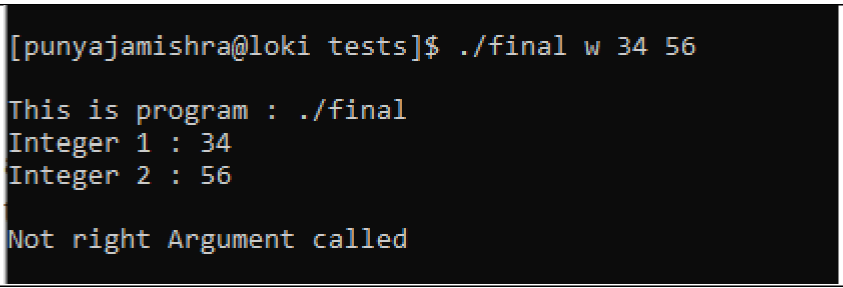
Just 1 arguments passed, we need 3

Test 3	
Input	Too many arguments passed
Expected Output	Prints error

Actual Output	
	`[punyajamishra@loki tests]$ ./final e 3 4 5 6`
	`This is program : ./final`
	`Not right number of arguments passed. Only needed 3`

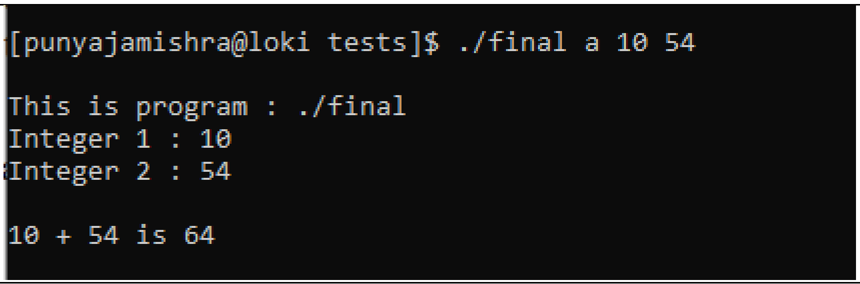
5 arguments passed we don’t need that many

Test 4	
Input	Wrong character passed – w
Expected Output	Prints error

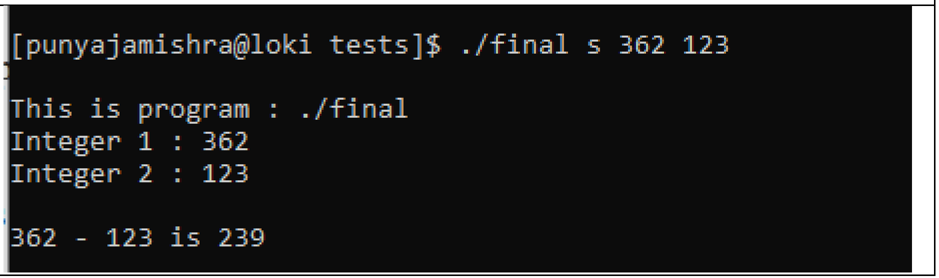
Actual Output	
	`[punyajamishra@loki tests]$ ./final w 34 56`
	
	`This is program : ./final`
	`Integer 1 : 34`
	`Integer 2 : 56`
	
	`Not right Argument called`

Wrong character – w – was passed. Valid characters are d, m, a, s

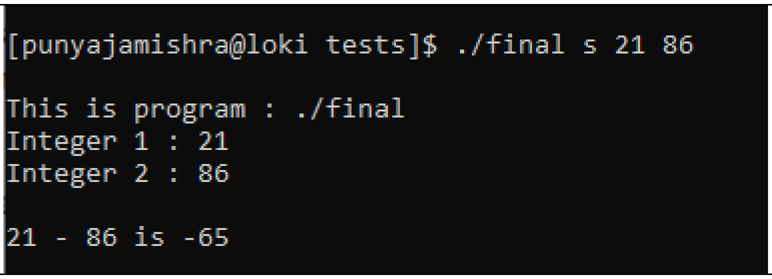
Test 5	
Input	Addition – a 10 54
Expected Output	10 + 54 is 64

Actual Output	
	`[punyajamishra@loki tests]$ ./final a 10 54`
	
	`This is program : ./final`
	`Integer 1 : 10`
	`Integer 2 : 54`
	
	`10 + 54 is 64`

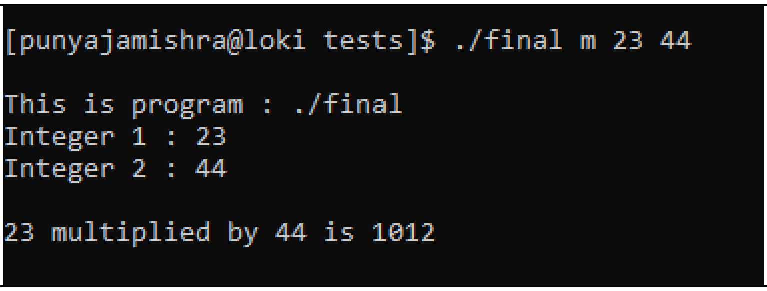
Now correct character a is passed and addition method prints the sum of the integers passed as well

Test 6	
Input	Subtraction – s 362 123
Expected Output	362 – 123 is 239
Actual Output	 [punyajamishra@loki tests]$ ./final s 362 123 This is program : ./final Integer 1 : 362 Integer 2 : 123 362 - 123 is 239

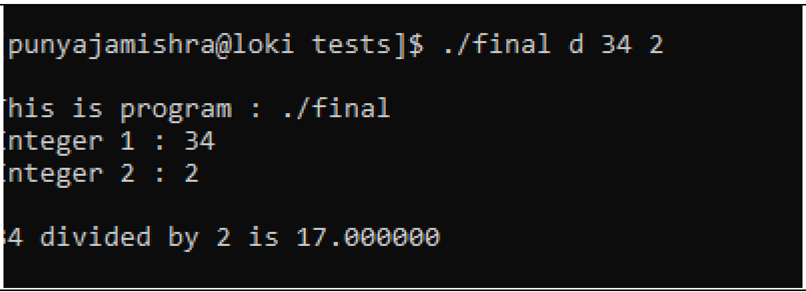
Now correct character s is passed and subtraction method prints the difference of the integers passed

Test 7	
Input	Subtraction – s 21 86
Expected Output	21 – 86 is -65
Actual Output	 [punyajamishra@loki tests]$ ./final s 21 86 This is program : ./final Integer 1 : 21 Integer 2 : 86 21 - 86 is -65

Now correct character s is passed and subtraction method prints the difference of the integers passed – this time negative output

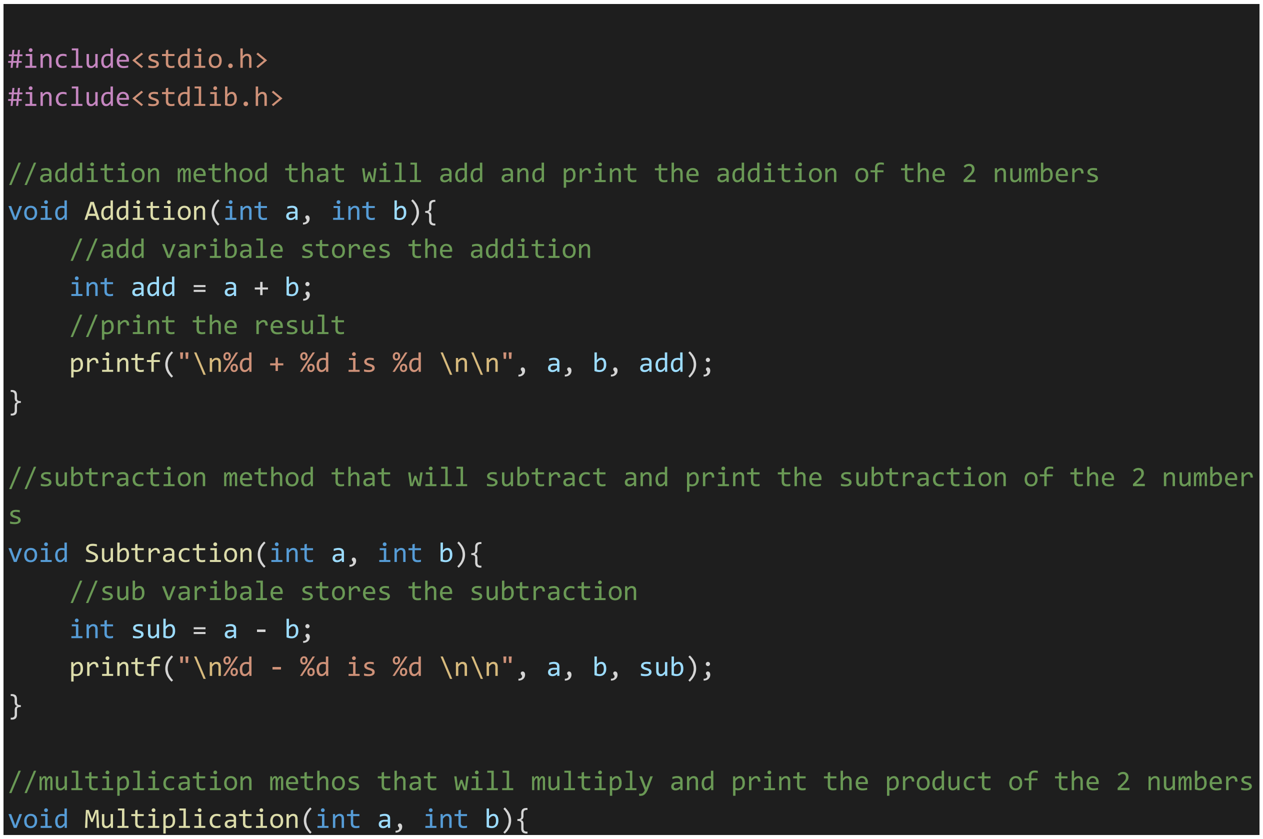
Test 8	
Input	Multiplication – m 23 44
Expected Output	23 multiplied by 44 is 1012
Actual Output	 [punyajamishra@loki tests]$ ./final m 23 44 This is program : ./final Integer 1 : 23 Integer 2 : 44 23 multiplied by 44 is 1012

Now correct character m is passed and multiplication method prints the product of the integers passed

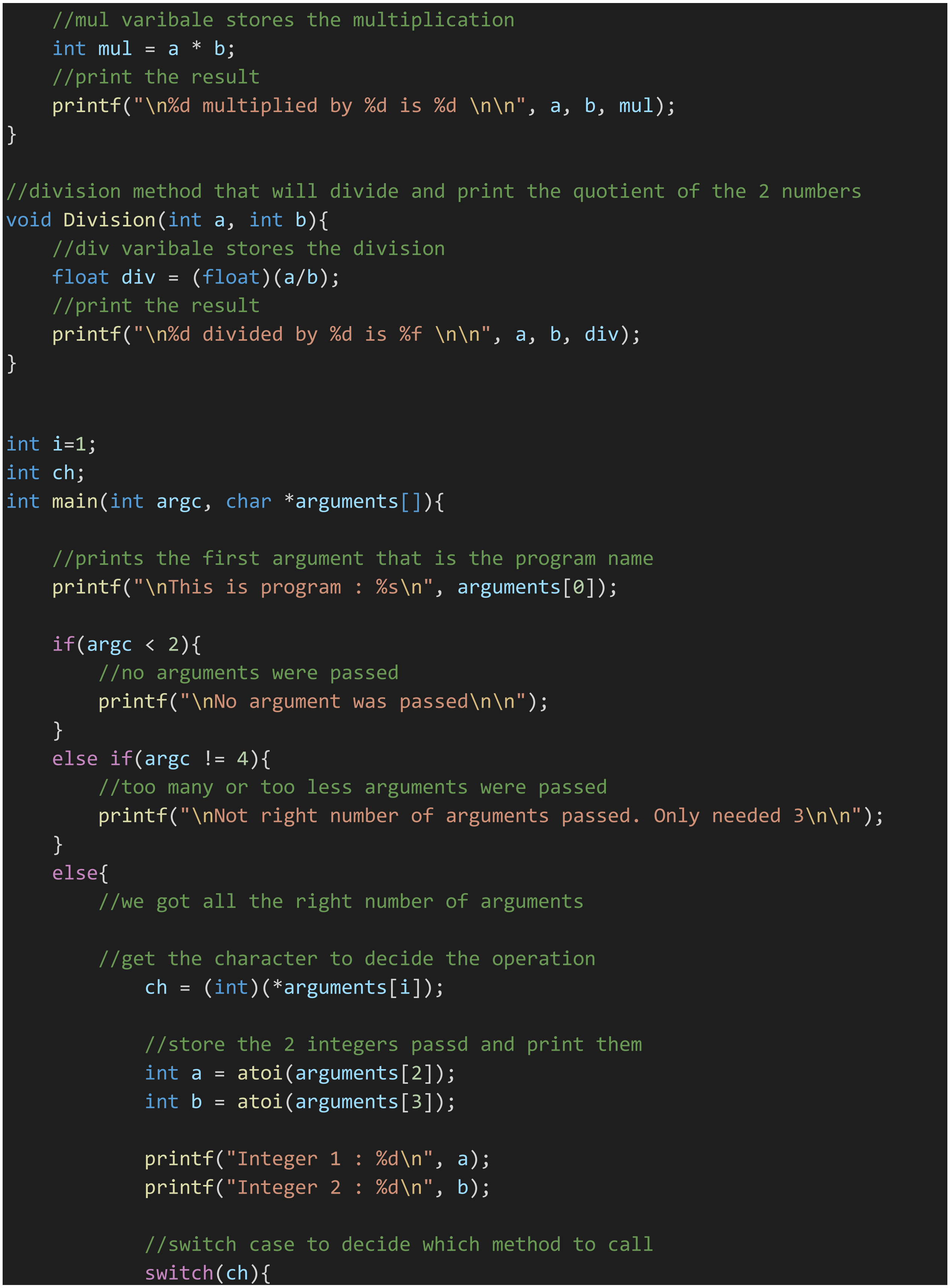
Test 9	
Input	Division - d 34 2
Expected Output	34 divided by 2 is 17.000
Actual Output	 punyajamishra@loki tests]$ ./final d 34 2 his is program : ./final nteger 1 : 34 nteger 2 : 2 4 divided by 2 is 17.000000

Now correct character d is passed and division method prints the quotient of the integers passed

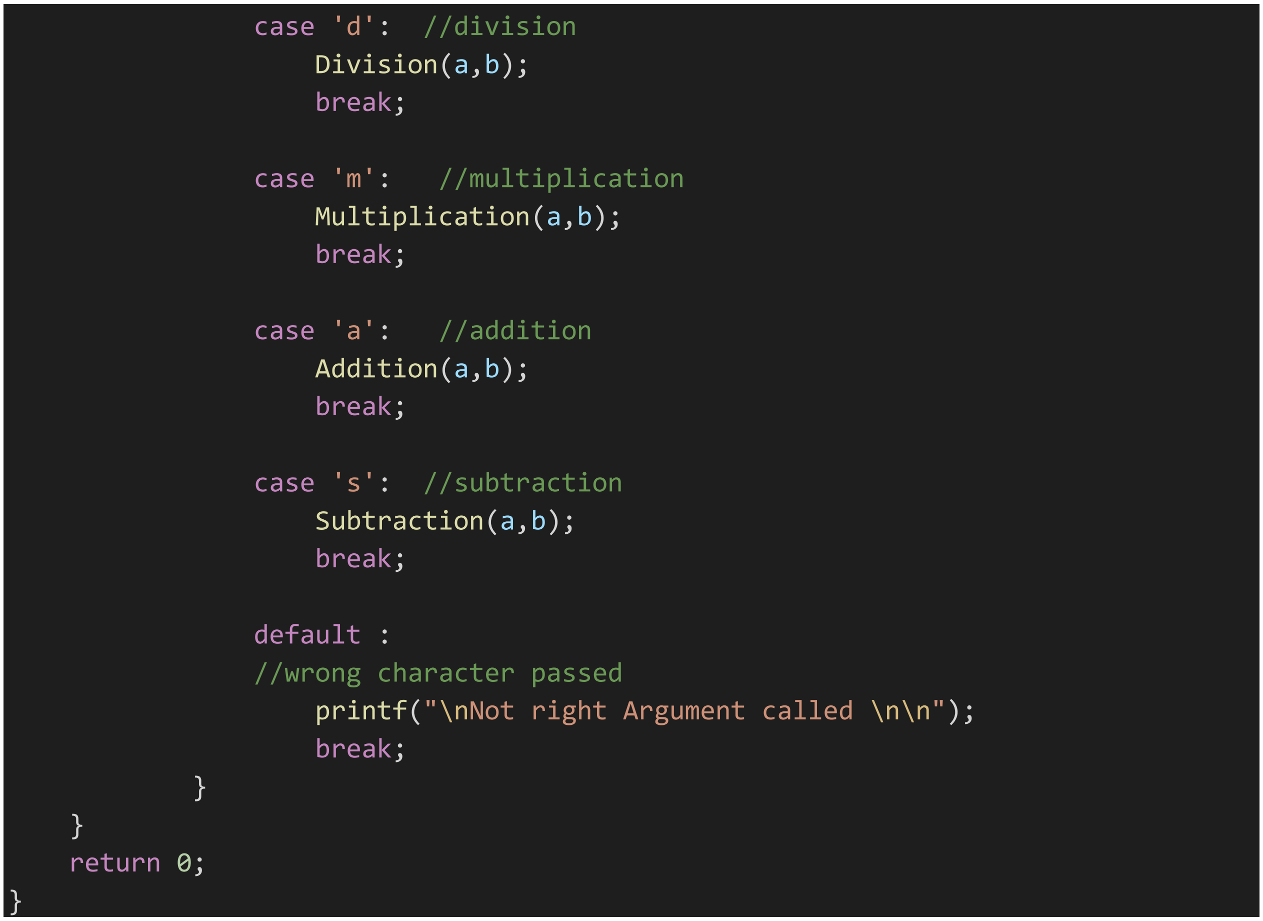
//code//



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

//addition method that will add and print the addition of the 2 numbers
void Addition(int a, int b){
    //add varibale stores the addition
    int add = a + b;
    //print the result
    printf("\n%d + %d is %d \n\n", a, b, add);
}

//subtraction method that will subtract and print the subtraction of the 2 number
s
void Subtraction(int a, int b){
    //sub varibale stores the subtraction
    int sub = a - b;
    printf("\n%d - %d is %d \n\n", a, b, sub);
}

//multiplication methos that will multiply and print the product of the 2 numbers
void Multiplication(int a, int b){
```



```
    //mul varibale stores the multiplication
    int mul = a * b;
    //print the result
    printf("\n%d multiplied by %d is %d \n\n", a, b, mul);
}

//division method that will divide and print the quotient of the 2 numbers
void Division(int a, int b){
    //div varibale stores the division
    float div = (float)(a/b);
    //print the result
    printf("\n%d divided by %d is %f \n\n", a, b, div);
}


int i=1;
int ch;
int main(int argc, char *arguments[]){

    //prints the first argument that is the program name
    printf("\nThis is program : %s\n", arguments[0]);

    if(argc < 2){
        //no arguments were passed
        printf("\nNo argument was passed\n\n");
    }
    else if(argc != 4){
        //too many or too less arguments were passed
        printf("\nNot right number of arguments passed. Only needed 3\n\n");
    }
    else{
        //we got all the right number of arguments

        //get the character to decide the operation
            ch = (int)(*arguments[i]);

            //store the 2 integers passd and print them
            int a = atoi(arguments[2]);
            int b = atoi(arguments[3]);

            printf("Integer 1 : %d\n", a);
            printf("Integer 2 : %d\n", b);

            //switch case to decide which method to call
            switch(ch){
```



```
                case 'd':  //division
                    Division(a,b);
                    break;

                case 'm':   //multiplication
                    Multiplication(a,b);
                    break;

                case 'a':   //addition
                    Addition(a,b);
                    break;

                case 's':  //subtraction
                    Subtraction(a,b);
                    break;

                default :
                //wrong character passed
                    printf("\nNot right Argument called \n\n");
                    break;
            }
    }
    return 0;
}
```