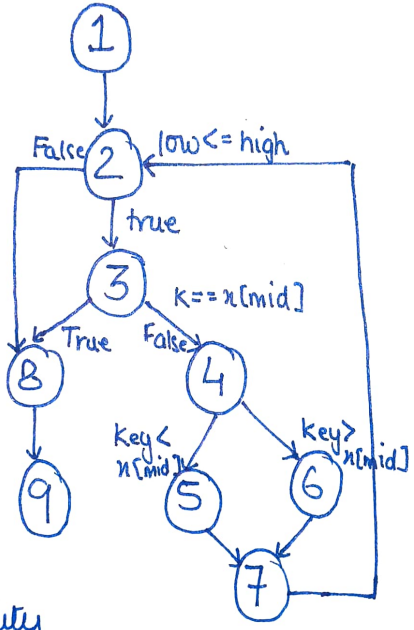
ASSIGNMENT

Q. Design and develop code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different test cases, execute these test case.

```
#include <stdio.h>
int binsrc(int x[], int low, int high, int key)
{
1  int mid;
2  while(low <= high)
3  {
4  mid = low+high/2;
5  if(x[mid] <= key)
6   low=mid+1;
7 else if(x[mid] >key)
8     high = mid-1;
9  else if (x[mid]==key)
10    return mid;
}}
int main(){
 int a[20], key, i, n, succ;
 printf("Enter the n value">
 scanf("%d",&key);
 succ=binser if(n>0)
 { printf("enter element in asc. order\n");
  for(i=0; i<n; i++)
    scanf("%d",&n);
    if(n>0)
      printf(enter key element);
    scanf("%d",&key);
    succ = binsrc(a,0,n-1, key);
```

```
    if (suc>=0)
    printf("Element found in position=%d\n",suc+1);
     else
      printf("Element not found");
   }
   else
    printf("No. should be greater than zero\n");
    return 0;
}
```



Cyclomatic complexity

$= E - N + 2P$

$= 11 - 9 + 2$

$= 4$

Independent path

P1: 1-2-3-8-9

P2: 1-2-3-4-5-7-2

P3: 1-2-3-4-6-7-2

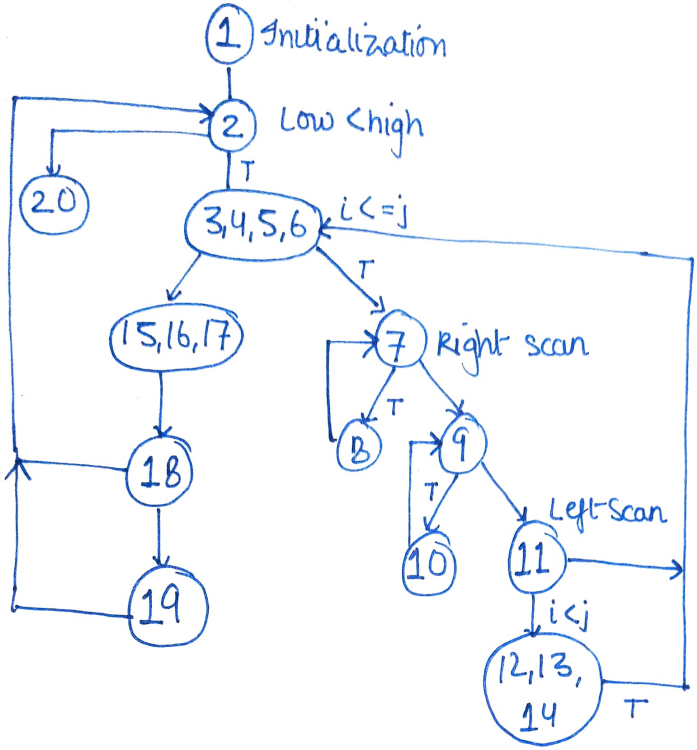
P4: 1-2-8-9

Test case Id	Summary	Pre-cond^n	Post-cond^n	Dependency	Input	Execution	Expected O/P	Actual O/P	Pass / Fail.
TC-1	1-2-3-8-9	Array in asc. order Array has ODD element	-	-	5 [10,20,30 40,50] 30	Enter n value Enter element Enter Key	Element found at 3		
TC-2	1-2-3-4-5-7-2	"	-	-	5 [10,20,30, 40,50] 20	Enter n value Enter element Enter Key	Element found at 2		
TC-3	1-2-3-4-6-7-2	"	-	-	5 [10,20,30, 40,50] 40	Enter n value Enter element Enter Key	Element found at 4		
TC-4	1-2-8-9	"	-	-	-5 Empty 2	Enter n value	No. should be greater than zero		
TC-4	1-2-8-9	"	-	-	5 [10,20,30, 40,50] 60	Enter n Enter element Enter Key	Element not found.		

ASSIGNMENT

Q. Design, develop and code and run the program in any suitable language to implement quick sort algorithm, Determine the basic path and using them derive different test case and execute these test cases.

```
Void quicksort (int x[10], int first, int last)
 {
1  int temp, pivot, i, j;
2  if (first < last)
3  {
4   pivot = first;
5   i = first;
6   j = last;
7   while (i<j)
8   {
9    while (x[i] <= x[pivot] && i<last)
10     i++;
11     while (x[i] > x[pivot])
12     j--;
13     if (i<j)
14     {
15      temp = x[i];
16      x[i] = x[j];
17      x[j] = temp;
18     }
19    temp = x[pivot];
20    x[pivot] = x[j];
21
22    x[j] = temp;
23    quicksort (x, first, j-1);
24    quicksort (x, j+1, last);
25    }
26 }
```



Cyclomatic complexity

$V(G) = E - N + 2P = 18 - 13 + 2 = 7$

Independent path

P1: A-B-N

P2 : A-B-C-J-K-B

P3: A-B-C-J-K-M-B

P4: A-B-C-D-F-H-C

P5 : A-B-C-D-F-H-I-C

P6: A-B-C-D-E-D-F-H

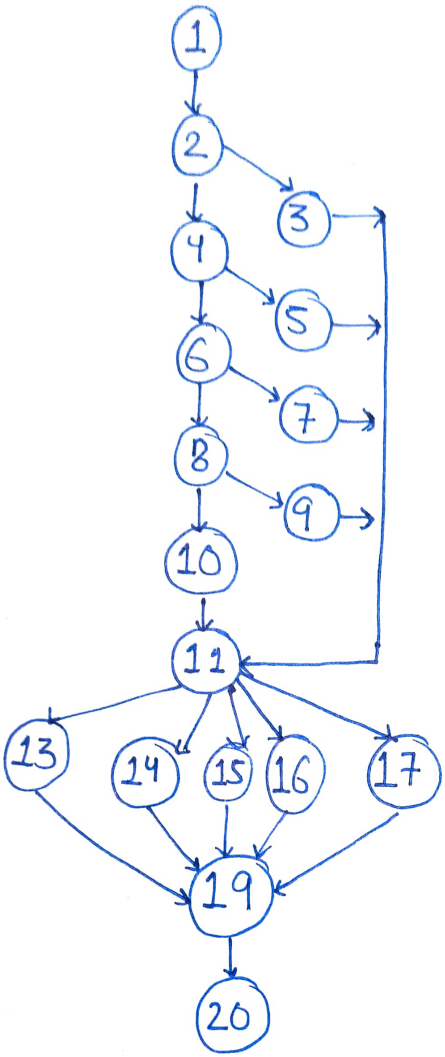
P7: A-B-C-D-F-G-F-H

Test case - Id	Summary	Pre - cond^{n}	Post - cond^{n}	Dependency	Input	Expected O/P	Actual O/P	Pass or fail
TC-1	Only 1 element	-	-	-	5 &	Sorted		
TC-2	Two elements	-	-	-	5,4	Repeated & Sorted		
TC-3	Three elements	-	-	-	1,2,3 or 3,2,1	Repeated & Sorted		
TC-4	Ascending sequence	-	-	-	1,2,3,4,5	Repeated & sorted		
TC-5	Descending Sequence	-	-	-	5,4,3,2,1	Repeated & Sorted		
TC-6	Pivot is min	-	-	-	1,4,3,2,5 or 2,2,2,2,2	Repeated and sorted		
TC-7	Pivot is max	-	-	-	5,2,3,1,4	Repeated and sorted.		

ASSIGNMENT

Q. Design, develop, code and run the program in any suitable language to implement an absolute letter grading procedure, making suitable assumption. Determine the basis paths and using them derive different test cases.

```
#include<stdio.h>
int main()
{
float per;
char grade;
scanf("%f",&per);
if(per>=90)
   grade='A';
else if(per>=80&&per<90)
   grade='B';
else if(per>=70&&per<80)
   grade='C';
else if(per>=60 && per<70)
   grade='D';
else grade='E';
switch(grade)
{
case 'A': printf("\nEXCELLENT"); break;
 case 'B': printf("\nVERY GOOD"); break;
 case 'C': printf("\nGOOD"); break;
 case 'D': printf("\nAbove average"); break;
 case 'E': printf("\nSatisfactory"); break;
}
printf("The percentage is %f and grade is %c", per, grade);
return 0;
}
```



cyclomatic complexity

$V(G) = E - N + 2P = 25 - 18 + 2 = 09$

independent path

P1: 1-2-4-6-8-10-11-17-19-20

P2: 1-2-4-6-8-9-11-16-19-20

P3: 1-2-4-6-7-11-15-19-20

P4: 1-2-4-5-11-14-19-20

P5: 1-2-3-11-13-19-20

P6: 1-2-4-6-B-10-11-13-19-20

P7: 1-2-4-6-B-10-11-14-19-20

P8: 1-2-4-6-B-10-11-15-19-20

P9: 1-2-4-6-B-10-11-16-19-20

Path	Input	Expected O/P	Remark
P1: 1-2-4-6-B-10-11-17-19-20	<60	E grade, satisfactory	
P2: 1-2-4-6-B-9-11-16-19-20	60-69	D Grade, Above average	
P3: 1-2-4-6-7-11-15-19-20	70-79	C Grade, Good	
P4: 1-2-4-5-11-14-19-20	80-89	B Grade, very good	
P5: 1-2-3-11-13-19-20	>=90	A grade, Excellent	
P6: 1-2-4-6-B-10-11-13-19-20	<60	Excellent	
P7: 1-2-4-6-B-10-11-14-19-20	<60	Very Good	
P8: 1-2-4-6-B-10-11-15-19-20	<60	Good	
P9: 1-2-4-6-B-10-11-16-19-20	<60	Above Average	

Q. Design and develope code and run the program in any suitable language to implement and solve the comission problem. Analyze it from the perspective of dataflow testing, derive different test cases and execute these test case and discuss result.

```
#include<stdio.h>
int main()
{
int locks, stocks, barrels, tlocks, tstocks, tbarrels;
float lprice, sprice, bprice, lsales, ssales, bsales, sales, comm;
lprice = 45.0;
sprice = 30.0;
bprice = 25.0;
tlocks = 0;
tstocks = 0;
tbarrels = 0;
printf("\n enter the no. of locks and to exit the loop enter -1 for locks \n");
scanf("%d", &locks);
while (locks != -1) {
printf("enter the number of stocks and barrels \n");
scanf("%d %d", &stocks, &barrels);
tlocks = tlocks + locks;
tstocks = tstocks + stocks;
tbarrels = tbarrels + barrels;
printf("\n enter the number of locks and to exit the loop enter -1 for locks \n"); scanf("%d", &locks);
}
printf("\n total locks = %d \n", tlocks);
printf("total stocks = %d \n", tstocks);
```

```
23  printf("total barrels = %d\n", tbarrels);
24  lsales = lprice * tlocks;
25  ssales = sprice * tstocks;
26  bsales = bprice * tbarrels;
27  sales = lsales + ssales + bsales;
28  printf("\n the total sales = %f\n", sales);
29  if(sales > 1800.0)
30  {
31  comm = 0.10 * 1000.0;
32  comm = comm + 0.15 * 800;
33  comm = comm + 0.20 * (sales - 1800.0);
34  }
35  else if(sales > 1000)
36  {
37  comm = 0.10 * 1000;
    comm = comm + 0.15 * (sales - 1000);
    }
38  else
39  comm = 0.10 * sales;
40  printf("the commission is = %f\n", comm);
41  return 0;
42  }
```

define/use node for variables in comission problem.

Variable name	Defined at node	Use at node
lprice	7	24
sprice	8	25
bprice	9	26
tlocks	10, 16	16, 21, 24
tstocks	11, 17	17, 22, 25
tbarrels	12, 18	18, 23, 26
locks	13, 19	14, 16
stocks	15	17
barrels	15	18
lsales	24	27
ssales	25	27
bsales	26	27
sales	27	28, 29, 33, 34, 37, 39
comm.	31, 32, 33, 36, 37, 39	32, 33, 37, 40

Test case Id	Description	Variable path	Du-path	Dc-path
TC-1	Check for lock price DEF(lprice, 7) & USE(lprice, 24)	(7,24)	<7-8-9-10-11-12 13-14-15-16-17- 18-19-20-21-22 -23-24>	Yes.
TC-2	Check for stock price DEF(sprice, 8) and USE(sprice, 25)	(8,25)	<8-9-10-11-12-13- 14-15-16-17-18- 19-20-21-22-23 -24-25>	Yes.
TC-3	Check for barrels price DEF(bprice, 9) and USE(bprice, 25)	(9,26)	<9-10-11-12-13-14 -15-16-17-18-19- 20-21-22-23-24- 25-26>	Yes.
TC-4	Check for total no. of locks DEF(tlocks, 16) and DEF(tlocks, 10) USE(tlocks, 18) USE(tlocks, 21) USE(tlocks, 24)	(10,26)	<10-11-12-13-14-15 -16>	Yes
		(10,21)	<10-11-12-13-14-15 -16-17-18-19-20-21>	No
		(10,24)	<10-11-12-13-14-15 -16-17-18-19-20-21-22-23-24>	No
		(16,16)	<16-16>	Yes
		(16-21)	<16-17-18-19-20- 14-21>	Yes.

Test case Id	Description	Variable path	Du-path	Dc-path
		(16,24)	<16-17-18-19-20-14-21-22-23-24>	No.
TC-5	Checks for total no. of stocks DEF(tstocks, 11) DEF(tstocks,17) USE(tstocks,17) USE(tstocks,22) USE(tstocks,25)	(11-17)	<11-12-13-14-15-16-17>	No
		(11,22)	<11-12-13-14-15-16-17-18-19-20-21-22>	No
		(11,25)	<11-12-13-14-15-16-17-18-19-20-21-22-23-24-25>	No
		(17,17)	<17;17>	Yes
		(17,22)	<17-18-19-20-14-21-22>	No
		(17,25)	<17-18-19-20-14-21-22-23-24-25>	No
TC-6	Checks for total no. of barrels DEF(tbarrels,12) DEF(tbarrels,18) USE(tbarrels,18) USE(tbarrels,23) USE(tbarrels,26)	(12-18)	<12-13-14-15-16-17-18>	No
		(12-23)	<12-13-14-15-16-17-18-19-20-14-21-22-23>	No

Variable name ID	Description	Variable path	Du-path	Dc-path
~~TC-1~~		(12-26)	<12-13-14-15-16-17-18-19-20-14-21-22-23-24-25-26>	No
		(18,18)	<18-18>	Yes
		(18,23)	<18-19-20-14-21-22-23>	~~Yes~~
		(18,26)	<18-19-20-14-21-22-23-24-25-26>	~~Yes~~
TC-7	Checks for locks DEF(locks, 13) DEF(locks, 19) USE(locks, 14) USE(locks, 16)	(13,14)	<13-14>	Yes
		(13-16)	<13-14-15-16>	Yes
		(19-14)	<19-20-14>	Yes
		(19-16)	<19-20-14-15-16>	Yes
TC-8	Checks for stocks DEF(locks, 15) USE(locks, 17)	(15-17)	(15-16-17>	Yes
TC-9	Check for barrels DEF(barrels, 15) USE(barrels, 18)	(15-18)	<15-16-17-18>	Yes

TEST CASE ID	Description	Variable path	Du-path	Dc-path
TC-10	Calculate sales of locks DEF(locks, 24) USE(locks, 27)	(24,27)	<24-25-26-27>	Yes.
TC-11	Calculate sales of stocks DEF(ssales, 25) USE(ssales, 27)	(25,27)	<25-26-27>	Yes
TC-12	Calculate sales of barrels DEF(bsales, 26) USE(bsales, 27)	(26,27)	<26-27>	Yes
TC-13	Calculate of total sale DEF(sale, 27) USE(sale, 28) USE(sale, 29) USE(sale, 33) USE(sale, 34) USE(sale, 37) USE(sale, 39)	(27,28) (27,29) (27-33> (27-34) (27-37) (27,39)	<27-28> <27-28-29> <27,28-29-30-31-32-33> <27-28-29-30-31-32-33-34> <27-28-29-30-31 32-33-34-35-36 37> <27-28-29-30-39>	Yes Yes Yes Yes Yes Yes

TEST CASE ID	Description	Variable path	D-u path	Dc path
TC-14	Calculation of commission earned	(31,32)	<31-32>	Yes
DEF(31) DEF(32)		(31,33)	<31-32-33>	Yes
DEF(comm,33)		(31,37)	<31-32-33-34-35-36-37>	Yes
DEF(comm,36)				
DEF(comm,37)		(31,40)	<31-32-33-34-35-36-37-38-39-40>	No Yes
DEF(comm,39)				
USE(comm,32)				
USE(comm,33)		(32,33,32 33,37,40)	<32-32>	Yes
USE(comm,37)			<32-33>	Yes
USE(comm,40)			<32-33-34-35-36-37>	Yes
			<32-33-34-35-36-37-39-40>	Yes
			<33-33>	Yes
			<33-34-35-36-37>	Yes
			<33-34-35-36-37-38-39-40>	Yes
		<37,37>	<37-37>	Yes
		<37,40>	<37-38-39-40>	Yes
		<39,40>	<39-40>	Yes