DSP LAB-1
Name : Satyam Kumar ID:20MCMB22

1. **Write a program to implement Tower of Hanoi without recursion. Also Debug the code using gdb debugger. Share the screenshots of the gdb execution part in your assignment.**

Compile file : gcc -g Q3.c -o Q3
Debugging : gdb ./Q3

I have created a breakpoint for the main function and moveSmallestDisk function
(gdb) b main
(gdb) b moveSmallestDisk
I have to run the object file generated during compilation of program
(gdb) r

Execution of Tower of Hanoi using GDB

1. It is given that main is at line 103, where the breakpoint was applied.
On reaching line 109, I was asked to input value of disks.

```
(gdb) b main
Breakpoint 1 at 0x55555555535f: file Q3.c, line 103.
(gdb) s
The program is not being run.
(gdb) n
The program is not being run.
(gdb) b moveSmallestDisk
Breakpoint 2 at 0x55555555516c: file Q3.c, line 11.
(gdb) s
The program is not being run.
(gdb) r
Starting program: /home/satyam/Public/UoH/DSPLab/Q3

Breakpoint 1, main () at Q3.c:103
103     {
(gdb) s
105         printf ("Enter the number of disks present in a rod ");
(gdb) s
108         scanf ("%d", &Disks);
(gdb) n
Enter the number of disks present in a rod 3
```

Fig1: Taking Input

2.

Since I have entered the no of disks as 3. As we can see that disk is odd, there is jump from line to line 117. On line 122, for loop is entered, as we can see that line 122 is repeated 4 times, as on the 4th iteration loop exits 123 is repeated 3 times, it can enter 3 times.

```
114         if ( (Disks % 2) == 0 )
(gdb) s
117             flag = -1;
(gdb) s
120         int rod[Disks + 1];
(gdb) s
122         for ( int i = 1; i <= Disks; ++i )
(gdb) s
123             rod[i] = 1;
(gdb) s
122         for ( int i = 1; i <= Disks; ++i )
(gdb) s
123             rod[i] = 1;
(gdb) s
122         for ( int i = 1; i <= Disks; ++i )
(gdb) s
123             rod[i] = 1;
(gdb) s
122         for ( int i = 1; i <= Disks; ++i )
(gdb) s
125         int moveCount = 0, disk, rodFrom, rodTo;
```

Fig2: Execution of Jump statement (if) and for loop

3. Execution of destinationCount function in while. For loop is executed 3 times as expected (line 94-95). Code jumps from line 128 to line 92, where the function is present.

```
128         while ( destinationCount (rod, Disks) != Disks )
(gdb) s
destinationCount (rod=0x7fffffffde70, numberOfDisks=3) at Q3.c:92
92          int count = 0;
(gdb) s
94          for ( int i = 1; i <= numberOfDisks; ++i ) {
(gdb) s
95              if ( rod[i] == 3 )
(gdb) s
94          for ( int i = 1; i <= numberOfDisks; ++i ) {
(gdb) s
95              if ( rod[i] == 3 )
(gdb) s
94          for ( int i = 1; i <= numberOfDisks; ++i ) {
(gdb) s
95              if ( rod[i] == 3 )
(gdb) s
94          for ( int i = 1; i <= numberOfDisks; ++i ) {
(gdb) s
99          return count;
(gdb) s
100     }
(gdb) s
```

Fig3: Execution of function destinationCount

4. Inside the while loop, next instruction is executed.

MoveCount is incremented. If statement is executed

(gdb) p moveCount

$1 = 1

Since movecount now is odd, smallestDisk function is executed (Jump from line 134 to line 11)

```
128         while ( destinationCount (rod, Disks) != Disks )
(gdb) s
131             ++moveCount;
(gdb) s
133             if ( (moveCount % 2) == 1 )
(gdb) s
134                 disk = moveSmallestDisk (rod, flag, &rodFrom, &rodTo);
(gdb) s
moveSmallestDisk (rod=0x7fffffffde70, flag=-1, src=0x7fffffffde90, dest=0x7ffff
fffde94) at Q3.c:11
11          *src = rod[1];  //1st rod is considered as src
(gdb) s
13          *dest = *src + flag; //Last rod is considered as dest
(gdb) s
17          if ( *dest > 3 )
(gdb) s
19          if ( *dest < 1 )
(gdb) s
20              *dest = 3;
(gdb) s
```

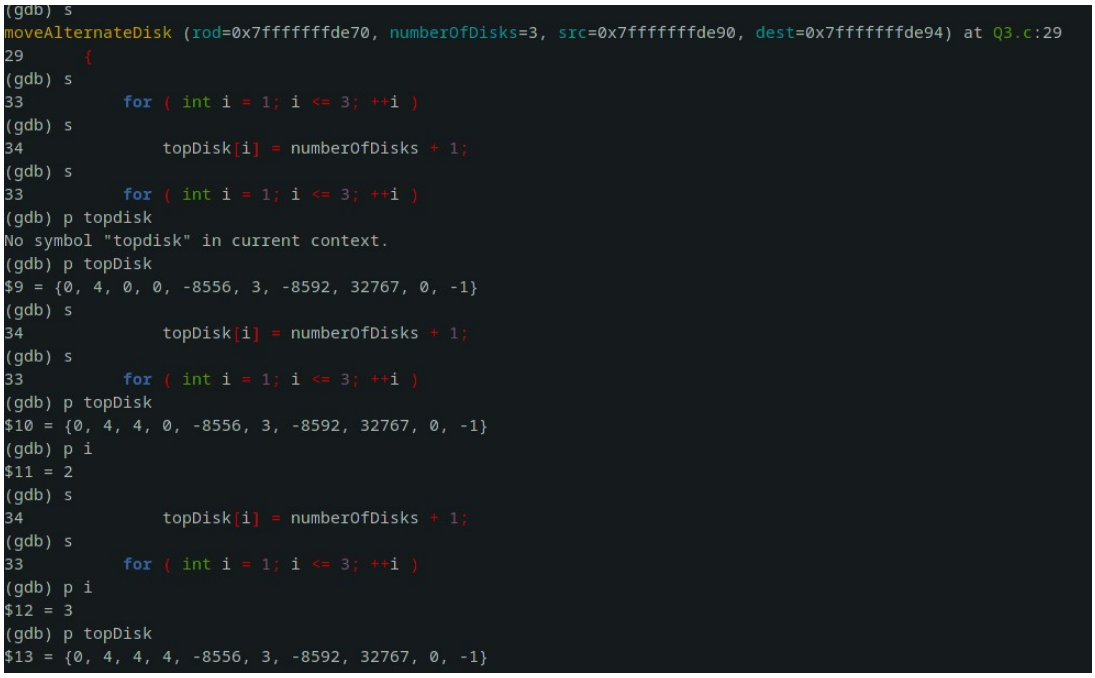
5. Print the statement (line 139 is executed)

Output 1: disk 1 rod A to rod C

```
main () at Q3.c:139
139             printf ("%i: disk %i rod %c to rod %c\n", moveCount, disk,
(gdb) s
1: disk 1 rod A to rod C
142             rod[disk] = rodTo;
```

6. While loop executes again Code jumps from line 128 to line 92, where the function is present.

7. Increment of movecount to 2. moveCount%2==0, moveAlternatedisk function is executed. (Jump from line 136 to line 29). On the entry to function, for loop is executed for 3 times.
If you want to know the values stored in topDisk, p topDisk is used, on each iteration of for loop.



8. After execution for loop (determine which disk is at the top of each rod), next instruction of for loop (line 37) begins.

```
37          for ( int i = numberOfDisks; i >= 1; --i )
(gdb) p i
$14 = 0
(gdb) s
38              topDisk[rod[i]] = i;
(gdb) p topDisk
$15 = {0, 4, 4, 4, -8556, 3, -8592, 32767, 0, -1}
(gdb) p topDisk[rod[0]]
$16 = 4
(gdb) s
37          for ( int i = numberOfDisks; i >= 1; --i )
(gdb) s
38              topDisk[rod[i]] = i;
(gdb) s
37          for ( int i = numberOfDisks; i >= 1; --i )
(gdb) s
38              topDisk[rod[i]] = i;
(gdb) s
37          for ( int i = numberOfDisks; i >= 1; --i )
(gdb) s
```

9. Switch statement is executed (line 41), after completion of for loop. Since rod 3 is already occupied. Only free rod left is rod 2. Next movement of disk takes place from rod 1 to rod 2.

```
41          switch (rod[1])
(gdb) s
66                  *src = 1;
(gdb) s
67                  *dest = 2;
(gdb) s
68                  break;
(gdb) s
76          if ( topDisk[*src] > topDisk[*dest] ) {
(gdb) p *src
$17 = 1
(gdb) p *dest
$18 = 2
```

10. Incremented moveCount to 3. moveSmallestDisk function is executed. Disk moves from rod 3 to rod 2, dest is 3. The disk moves rod 3 to rod 2.

```
142             rod[disk] = rodTo;
(gdb) n
128         while ( destinationCount (rod, Disks) != Disks )
(gdb) n
131             ++moveCount;
(gdb) n
133             if ( (moveCount % 2) == 1 )
(gdb) n
134                 disk = moveSmallestDisk (rod, flag, &rodFrom, &rodTo);
(gdb) n
139             printf ("%i: disk %i rod %c to rod %c\n", moveCount, disk,
(gdb) n
3: disk 1 rod C to rod B
```

11. Incremented moveCount to 4. moveAlternateDisk function is executed. Now rod 3 is empty, rod 2 is not empty, the disk moves from rod 1 to rod3, which case 2 of switch statement

```
128         while ( destinationCount (rod, Disks) != Disks )
(gdb) n
131             ++moveCount;
(gdb) p moveCount
$20 = 3
(gdb) s
133             if ( (moveCount % 2) == 1 )
(gdb) p moveCount
$21 = 4
```

```
37          for ( int i = numberOfDisks; i >= 1; --i )
(gdb) s
41          switch (rod[1])
(gdb) s
57                  *src = 1;
(gdb) ss
sUndefined command: "ss".  Try "help".
(gdb) s
58                  *dest = 3;
(gdb) s
59                  break;
(gdb) s
76          if ( topDisk[*src] > topDisk[*dest] ) {
(gdb) s
85          return topDisk[*src];
(gdb) s
86      }
(gdb) s
main () at Q3.c:139
139             printf ("%i: disk %i rod %c to rod %c\n", moveCount, disk,
(gdb) s
4: disk 3 rod A to rod C
```

12. Incrmented moveCount to 5. Since it is odd, moveSmallestDisk function is executed. Initially *dest=3, then it is reset to 1.

```
131             ++moveCount;
(gdb) s
133             if ( (moveCount % 2) == 1 )
(gdb) s
134                 disk = moveSmallestDisk (rod, flag, &rodFrom, &rodTo);
(gdb) s
moveSmallestDisk (rod=0x7fffffffde70, flag=-1, src=0x7fffffffde90, dest=0x7fffffffde94) at Q3.c:11
11          *src = rod[1];  //1st rod is considered as src
(gdb) p *src
$24 = 1
(gdb) p moveCount
No symbol "moveCount" in current context.
(gdb) s
13          *dest = *src + flag; //Last rod is considered as dest
(gdb) p *dest
$25 = 3
(gdb) p flag
$26 = -1
```

```
17          if ( *dest > 3 )
(gdb) s
19          if ( *dest < 1 )
(gdb) p *dest
$27 = 1
(gdb) s
23          return 1;
(gdb) s
24      }
(gdb) s
main () at Q3.c:139
139             printf ("%i: disk %i rod %c to rod %c\n", moveCount, disk,
(gdb) s
5: disk 1 rod B to rod A
```

13. Incremented moveCount to 6. moveAlternateDisk function is executed.
Incremented moveCount to 7. moveSmallestDisk function is executed.

```
128         while ( destinationCount (rod, Disks) != Disks )
(gdb) s
destinationCount (rod=0x7fffffffde70, numberOfDisks=3) at Q3.c:92
92          int count = 0;
(gdb) p count
$28 = -8592
(gdb) c
Continuing.
6: disk 2 rod B to rod C
7: disk 1 rod A to rod C
```