Registration Number: 19BCE2119

Name: Gaurav Kumar Singh

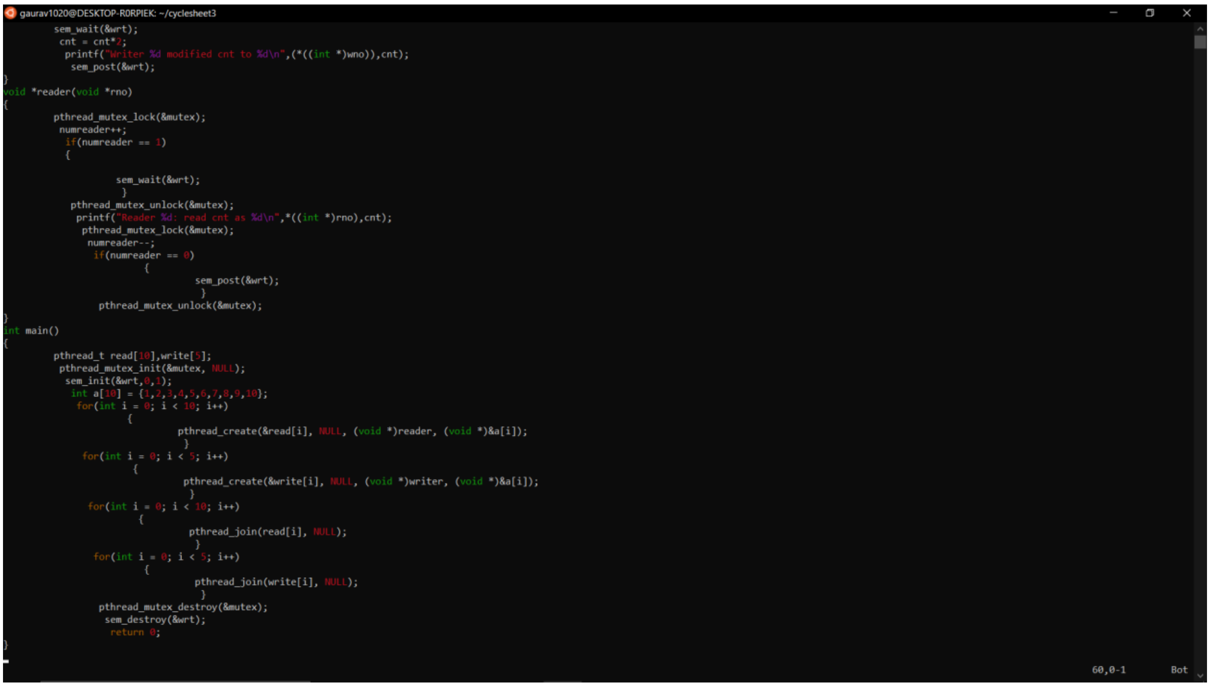
Cyclesheet-3

15. Write a program to provide a solution for reader- writer problem / producer consumer using semaphore.

CODE

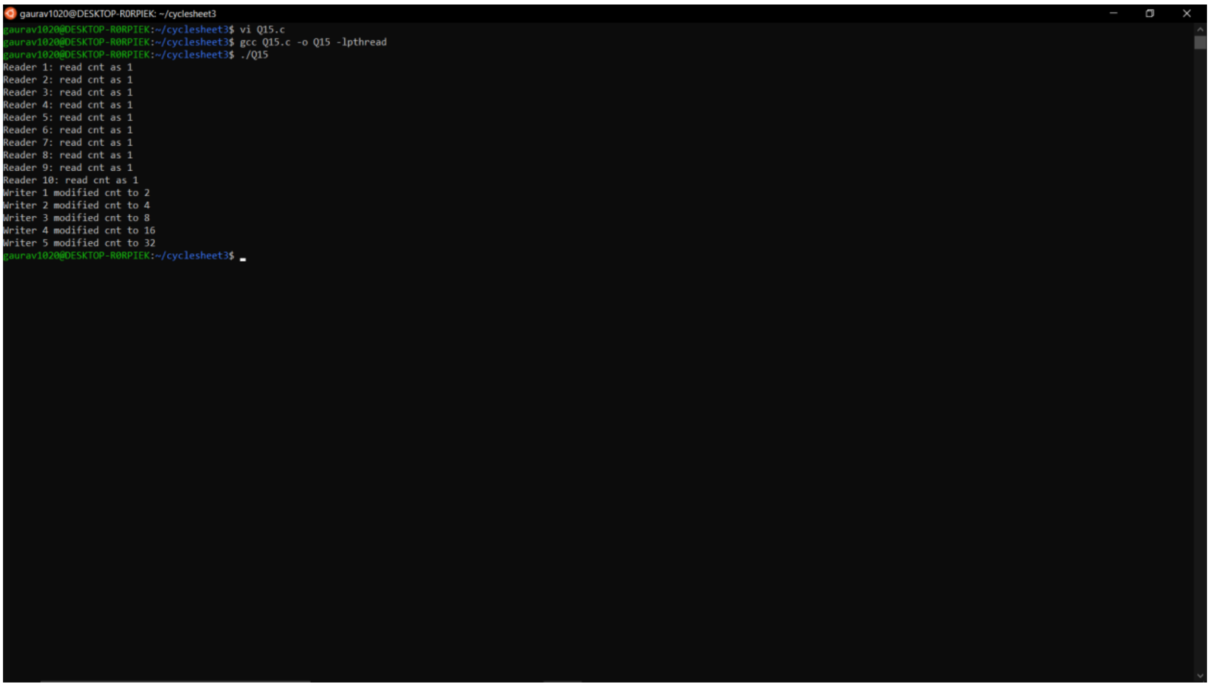
```
#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

sem_t wrt;

pthread_mutex_t mutex;

int cnt = 1;

int numreader = 0;

void *writer(void *wno)

{

        sem_wait(&wrt);

         cnt = cnt*2;

          printf("Writer %d modified cnt to %d\n",(*((int *)wno)),cnt);

           sem_post(&wrt);

}

void *reader(void *rno)

{

        pthread_mutex_lock(&mutex);

         numreader++;

          if(numreader == 1)

          {


                   sem_wait(&wrt);

                    }
        pthread_mutex_unlock(&mutex);
         printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
          pthread_mutex_lock(&mutex);
           numreader--;
            if(numreader == 0)
                  {
                        sem_post(&wrt);
                         }
             pthread_mutex_unlock(&mutex);
}
int main()
{
      pthread_t read[10],write[5];
       pthread_mutex_init(&mutex, NULL);
        sem_init(&wrt,0,1);
         int a[10] = {1,2,3,4,5,6,7,8,9,10};
          for(int i = 0; i < 10; i++)
                {
                      pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
                       }
           for(int i = 0; i < 5; i++)
                 {
                       pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
                        }
            for(int i = 0; i < 10; i++)
                  {
                        pthread_join(read[i], NULL);
                         }
             for(int i = 0; i < 5; i++)
                   {
                         pthread_join(write[i], NULL);
                         }
             pthread_mutex_destroy(&mutex);
              sem_destroy(&wrt);
               return 0;
}
```

OUTPUT



gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3

```
         sem_wait(&wrt);
          cnt = cnt*2;
           printf("Writer %d modified cnt to %d\n",(*((int *)wno)),cnt);
            sem_post(&wrt);
}
void *reader(void *rno)
{
         pthread_mutex_lock(&mutex);
          numreader++;
           if(numreader == 1)
           {

                    sem_wait(&wrt);
                     }
            pthread_mutex_unlock(&mutex);
             printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
              pthread_mutex_lock(&mutex);
               numreader--;
                if(numreader == 0)
                         {
                                  sem_post(&wrt);
                                   }
                 pthread_mutex_unlock(&mutex);
}
int main()
{
         pthread_t read[10],write[5];
          pthread_mutex_init(&mutex, NULL);
           sem_init(&wrt,0,1);
            int a[10] = {1,2,3,4,5,6,7,8,9,10};
             for(int i = 0; i < 10; i++)
                      {
                               pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
                                }
              for(int i = 0; i < 5; i++)
                       {
                                pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
                                 }
               for(int i = 0; i < 10; i++)
                        {
                                 pthread_join(read[i], NULL);
                                  }
                for(int i = 0; i < 5; i++)
                         {
                                  pthread_join(write[i], NULL);
                                   }
                 pthread_mutex_destroy(&mutex);
                  sem_destroy(&wrt);
                   return 0;
}
_
                                                                                        60,0-1        Bot
```



gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3

```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ vi Q15.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ gcc Q15.c -o Q15 -lpthread
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ ./Q15
Reader 1: read cnt as 1
Reader 2: read cnt as 1
Reader 3: read cnt as 1
Reader 4: read cnt as 1
Reader 5: read cnt as 1
Reader 6: read cnt as 1
Reader 7: read cnt as 1
Reader 8: read cnt as 1
Reader 9: read cnt as 1
Reader 10: read cnt as 1
Writer 1 modified cnt to 2
Writer 2 modified cnt to 4
Writer 3 modified cnt to 8
Writer 4 modified cnt to 16
Writer 5 modified cnt to 32
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ _
```

16. Implement a solution for the classical synchronization problem: Dining Philosophers.

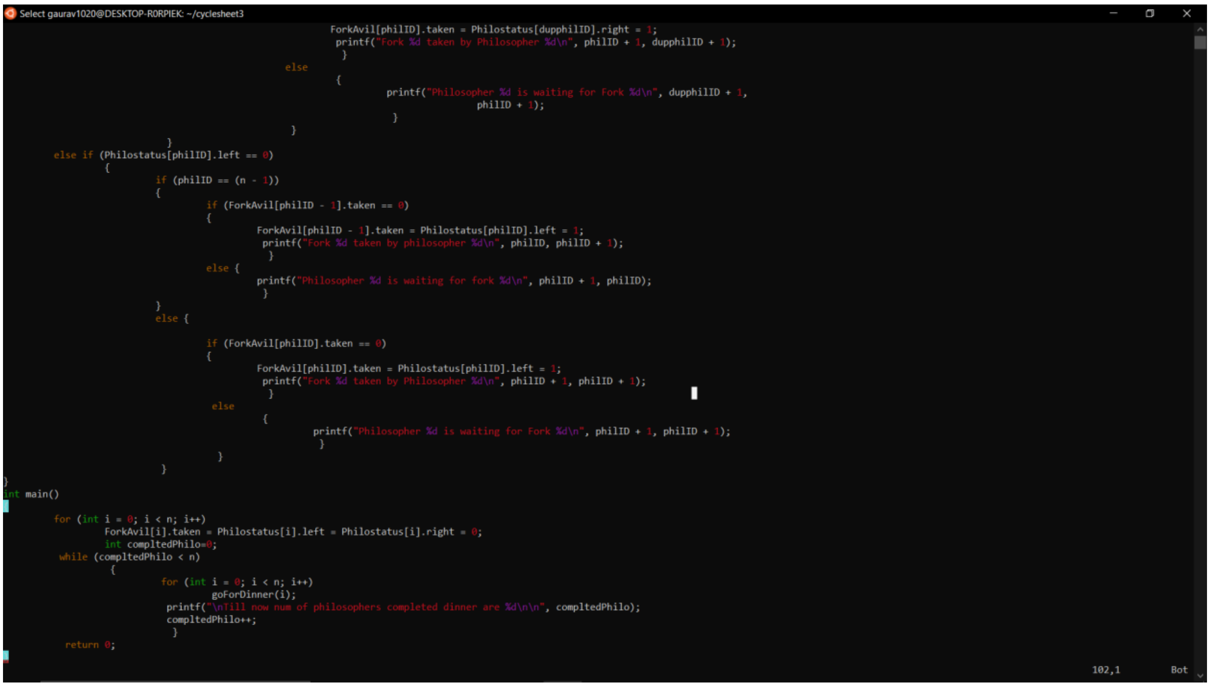
CODE

```
#include<stdio.h>

#define n 5

int compltedPhilo = 0, i;

struct fork

{

    int taken;

}

ForkAvil[n];

struct philosp

{

    int left;

     int right;

}

Philostatus[n];

void goForDinner(int philID)

{

    if (Philostatus[philID].left == 10 && Philostatus[philID].right == 10)

        printf("Philosopher %d completed his dinner\n", philID + 1);

    else if (Philostatus[philID].left == 1 && Philostatus[philID].right == 1)

        {

            printf("Philosopher %d completed his dinner\n", philID + 1);

             Philostatus[philID].left = Philostatus[philID].right = 10;

              int otherFork = philID - 1;

               if (otherFork == -1)

                   otherFork = (n - 1);

                ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;

                 printf("Philosopher %d released fork %d and fork %d\n", philID + 1, philID + 1, otherFork + 1);

                  compltedPhilo++;
```

```
                        }
          else if (Philostatus[philID].left == 1 && Philostatus[philID].right == 0)
               {
                    if (philID == (n - 1))
                    {
                         if (ForkAvil[philID].taken == 0)
                         {
                              ForkAvil[philID].taken = Philostatus[philID].right = 1;
                              printf("Fork %d taken by philosopher %d\n", philID + 1, philID + 1);
                              }
                          else
                              {
                                   printf("Philosopher %d is waiting for fork %d\n", philID+1, philID + 1);
                                   }
                          }
                    else
                         {
                              int dupphilID = philID;
                              philID -= 1;
                               if (philID == -1)
                                   philID = (n - 1);
                               if (ForkAvil[philID].taken == 0)
                               {
                                    ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
                                   printf("Fork %d taken by Philosopher %d\n", philID + 1, dupphilID + 1);
                                     }
                                else
                                     {
                                          printf("Philosopher %d is waiting for Fork %d\n", dupphilID +
1,
                                                  philID + 1);
                              }
                       }
             }
else if (Philostatus[philID].left == 0)
        {
            if (philID == (n - 1))
            {
                if (ForkAvil[philID - 1].taken == 0)
                {
                    ForkAvil[philID - 1].taken = Philostatus[philID].left = 1;
                     printf("Fork %d taken by philosopher %d\n", philID, philID + 1);
                      }
                else {
                    printf("Philosopher %d is waiting for fork %d\n", philID + 1, philID);
                     }
            }
            else {


                if (ForkAvil[philID].taken == 0)
                {
                    ForkAvil[philID].taken = Philostatus[philID].left = 1;
                     printf("Fork %d taken by Philosopher %d\n", philID + 1, philID + 1);
                      }
                 else
                     {
                        printf("Philosopher %d is waiting for Fork %d\n", philID + 1, philID +
1);
                         }
                  }
             }
}
```

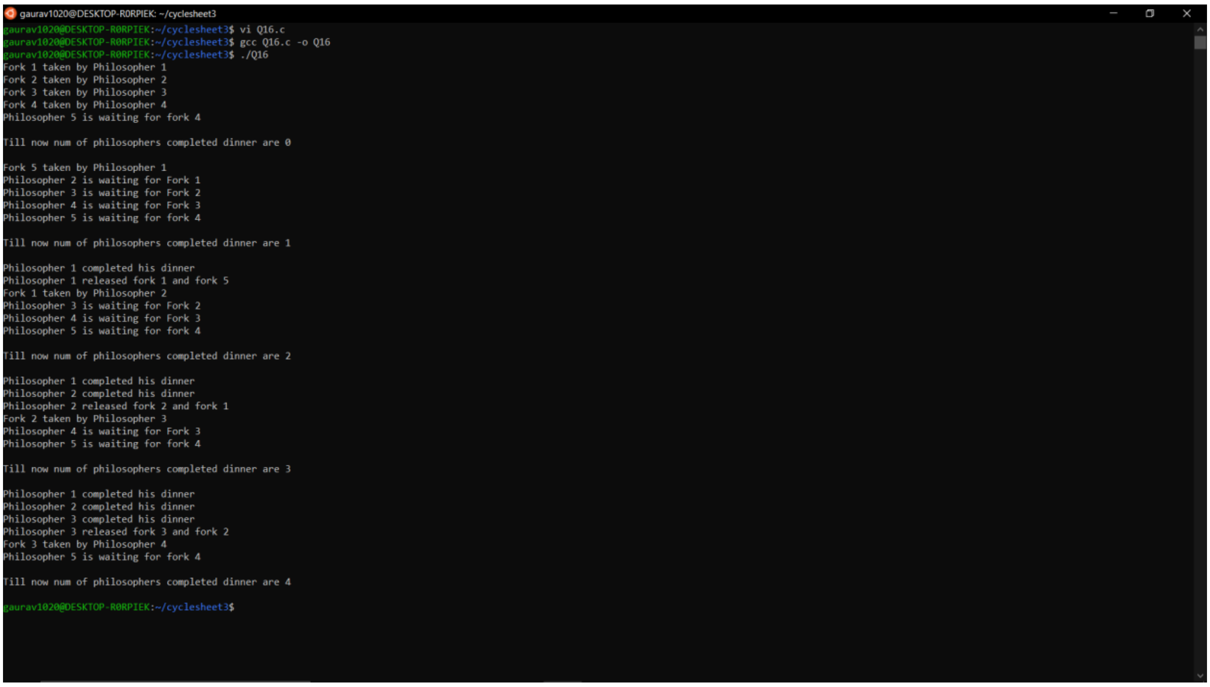
```
int main()

{

        for (int i = 0; i < n; i++)

                ForkAvil[i].taken = Philostatus[i].left = Philostatus[i].right = 0;

                int compltedPhilo=0;

         while (compltedPhilo < n)

                 {

                         for (int i = 0; i < n; i++)

                                 goForDinner(i);

                          printf("\nTill now num of philosophers completed dinner are %d\n\n",
compltedPhilo);

                          compltedPhilo++;

                           }

          return 0;

}
```

OUTPUT



Select gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3

```
                                                ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
                                                 printf("Fork %d taken by Philosopher %d\n", philID + 1, dupphilID + 1);
                                                  }
                                        else
                                                 {
                                                         printf("Philosopher %d is waiting for Fork %d\n", dupphilID + 1,
                                                                         philID + 1);
                                                          }
                                         }
                        }
        else if (Philostatus[philID].left == 0)
                {
                        if (philID == (n - 1))
                        {
                                if (ForkAvil[philID - 1].taken == 0)
                                {
                                        ForkAvil[philID - 1].taken = Philostatus[philID].left = 1;
                                         printf("Fork %d taken by philosopher %d\n", philID, philID + 1);
                                          }
                                else {
                                        printf("Philosopher %d is waiting for fork %d\n", philID + 1, philID);
                                         }
                        }
                        else {

                                if (ForkAvil[philID].taken == 0)
                                {
                                        ForkAvil[philID].taken = Philostatus[philID].left = 1;
                                         printf("Fork %d taken by Philosopher %d\n", philID + 1, philID + 1);
                                          }
                                 else
                                         {
                                                 printf("Philosopher %d is waiting for Fork %d\n", philID + 1, philID + 1);
                                                  }
                                 }
                         }
}
int main()
{
        for (int i = 0; i < n; i++)
                ForkAvil[i].taken = Philostatus[i].left = Philostatus[i].right = 0;
                int compltedPhilo=0;
         while (compltedPhilo < n)
                 {
                         for (int i = 0; i < n; i++)
                                 goForDinner(i);
                          printf("\nTill now num of philosophers completed dinner are %d\n\n", compltedPhilo);
                          compltedPhilo++;
                           }
          return 0;
}
                                                                                                    102,1          Bot
```



```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ vi Q16.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ gcc Q16.c -o Q16
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ ./Q16
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Fork 4 taken by Philosopher 4
Philosopher 5 is waiting for fork 4

Till now num of philosophers completed dinner are 0

Fork 5 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for Fork 3
Philosopher 5 is waiting for fork 4

Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 5
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for Fork 3
Philosopher 5 is waiting for fork 4

Till now num of philosophers completed dinner are 2

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for Fork 3
Philosopher 5 is waiting for fork 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by Philosopher 4
Philosopher 5 is waiting for fork 4

Till now num of philosophers completed dinner are 4

gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$
```

17. Write a program to avoid deadlock using Banker's algorithm.(Safety algorithm)

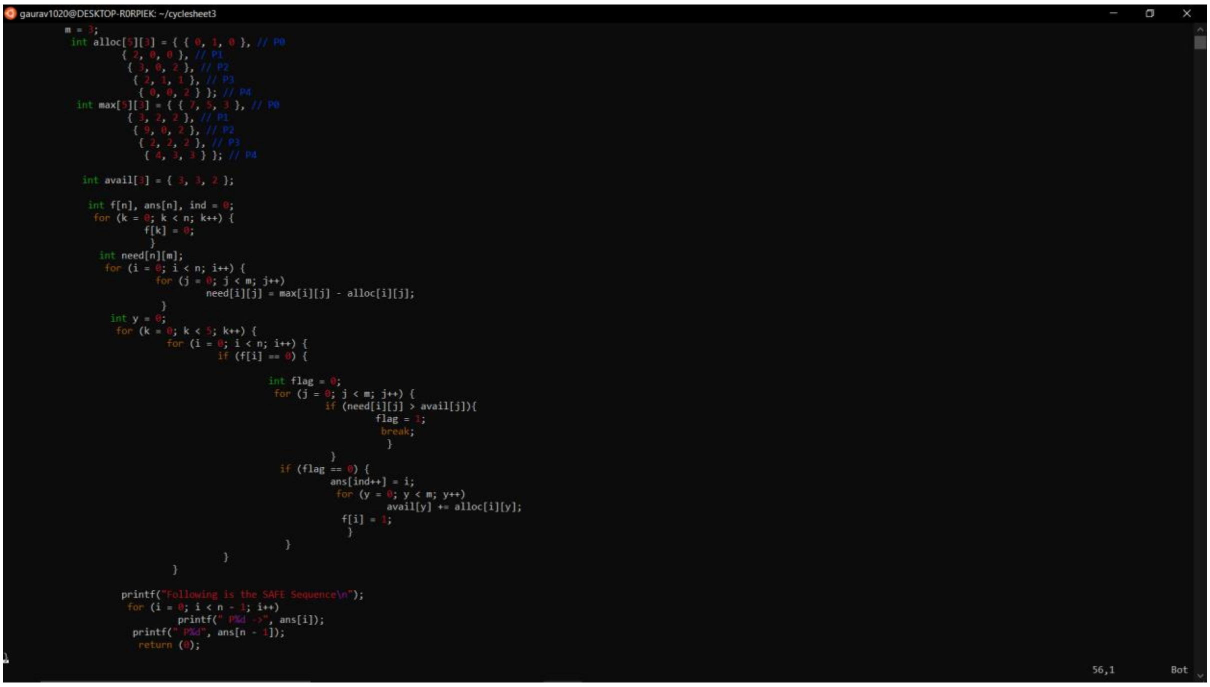
CODE

```
#include <stdio.h>

int main()

{

        int n, m, i, j, k;

         n = 5;

          m = 3;

           int alloc[5][3] = { { 0, 1, 0 }, // P0

                   { 2, 0, 0 }, // P1

                    { 3, 0, 2 }, // P2

                     { 2, 1, 1 }, // P3

                      { 0, 0, 2 } }; // P4

            int max[5][3] = { { 7, 5, 3 }, // P0

                    { 3, 2, 2 }, // P1

                     { 9, 0, 2 }, // P2

                      { 2, 2, 2 }, // P3

                       { 4, 3, 3 } }; // P4
```

```
int avail[3] = { 3, 3, 2 };


 int f[n], ans[n], ind = 0;
  for (k = 0; k < n; k++) {
         f[k] = 0;
          }
   int need[n][m];
   for (i = 0; i < n; i++) {
           for (j = 0; j < m; j++)
                   need[i][j] = max[i][j] - alloc[i][j];
            }
    int y = 0;
     for (k = 0; k < 5; k++) {
             for (i = 0; i < n; i++) {
                    if (f[i] == 0) {

                            int flag = 0;
                             for (j = 0; j < m; j++) {
                                     if (need[i][j] > avail[j]){
                                            flag = 1;
                                             break;
                                              }
                                     }
                              if (flag == 0) {
                                     ans[ind++] = i;
                                      for (y = 0; y < m; y++)
                                              avail[y] += alloc[i][y];
                                       f[i] = 1;
                                        }
                               }
                     }
```

```
                }
printf("Following is the SAFE Sequence\n");
            for (i = 0; i < n - 1; i++)
                printf(" P%d ->", ans[i]);
             printf(" P%d", ans[n - 1]);
              return (0);
}
```

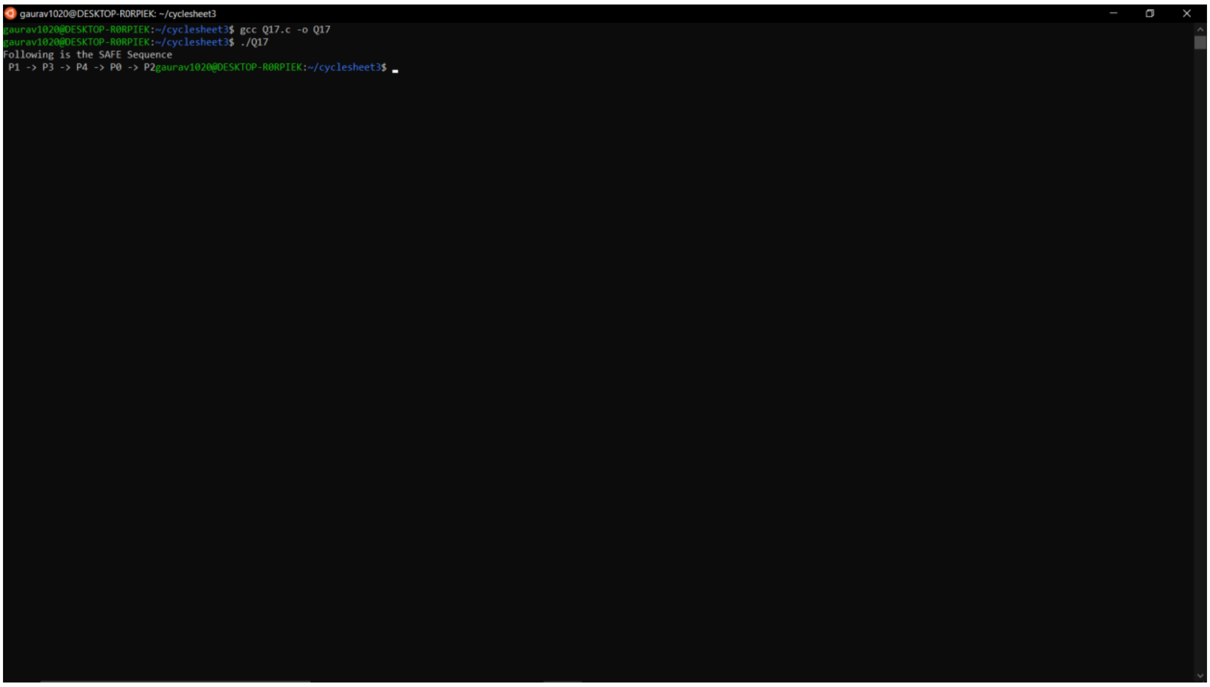
OUTPUT



gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3

```
          m = 3;
           int alloc[5][3] = { { 0, 1, 0 }, // P0
                    { 2, 0, 0 }, // P1
                     { 3, 0, 2 }, // P2
                      { 2, 1, 1 }, // P3
                       { 0, 0, 2 } }; // P4
            int max[5][3] = { { 7, 5, 3 }, // P0
                     { 3, 2, 2 }, // P1
                      { 9, 0, 2 }, // P2
                       { 2, 2, 2 }, // P3
                        { 4, 3, 3 } }; // P4

             int avail[3] = { 3, 3, 2 };

              int f[n], ans[n], ind = 0;
               for (k = 0; k < n; k++) {
                        f[k] = 0;
                         }
                int need[n][m];
                 for (i = 0; i < n; i++) {
                          for (j = 0; j < m; j++)
                                   need[i][j] = max[i][j] - alloc[i][j];
                           }
                  int y = 0;
                   for (k = 0; k < 5; k++) {
                            for (i = 0; i < n; i++) {
                                     if (f[i] == 0) {

                                              int flag = 0;
                                               for (j = 0; j < m; j++) {
                                                        if (need[i][j] > avail[j]){
                                                                 flag = 1;
                                                                  break;
                                                                   }
                                                         }
                                                if (flag == 0) {
                                                         ans[ind++] = i;
                                                          for (y = 0; y < m; y++)
                                                                   avail[y] += alloc[i][y];
                                                           f[i] = 1;
                                                            }
                                                 }
                                      }
                             }

                    printf("Following is the SAFE Sequence\n");
                     for (i = 0; i < n - 1; i++)
                              printf(" P%d ->", ans[i]);
                      printf(" P%d", ans[n - 1]);
                       return (0);
}
```

56,1 Bot



```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ gcc Q17.c -o Q17
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ ./Q17
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$
```

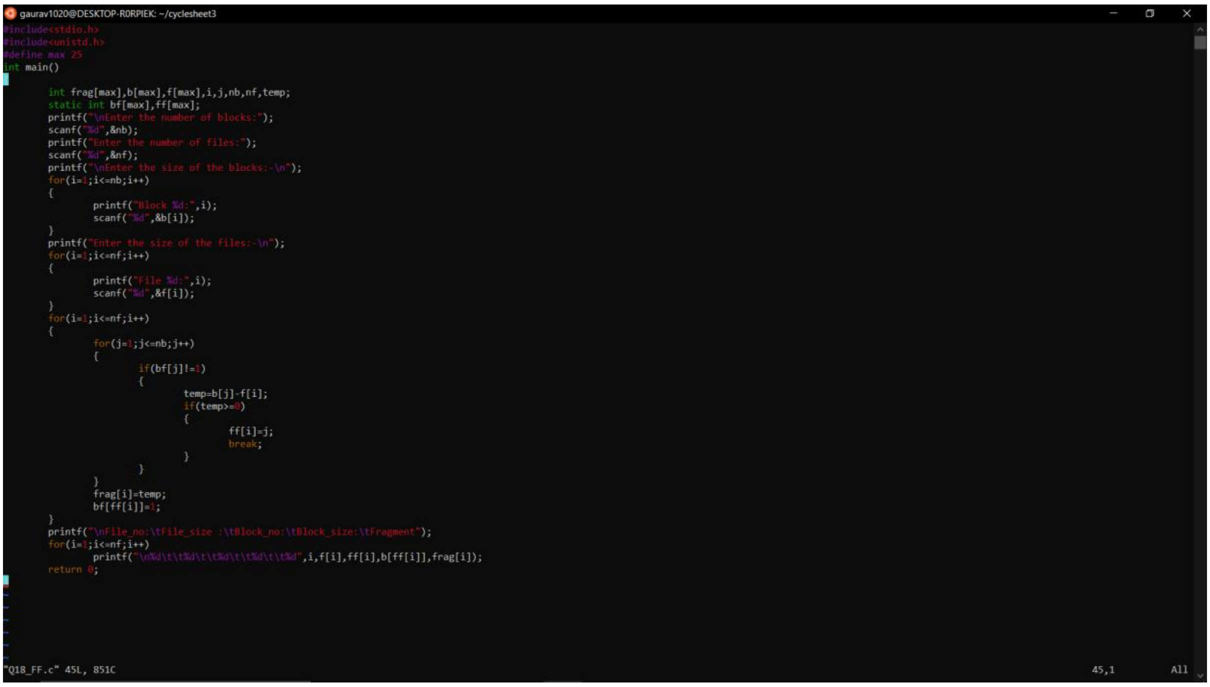
18. Implement a program to allocate memory by applying the following strategies.

a. FIRST FIT

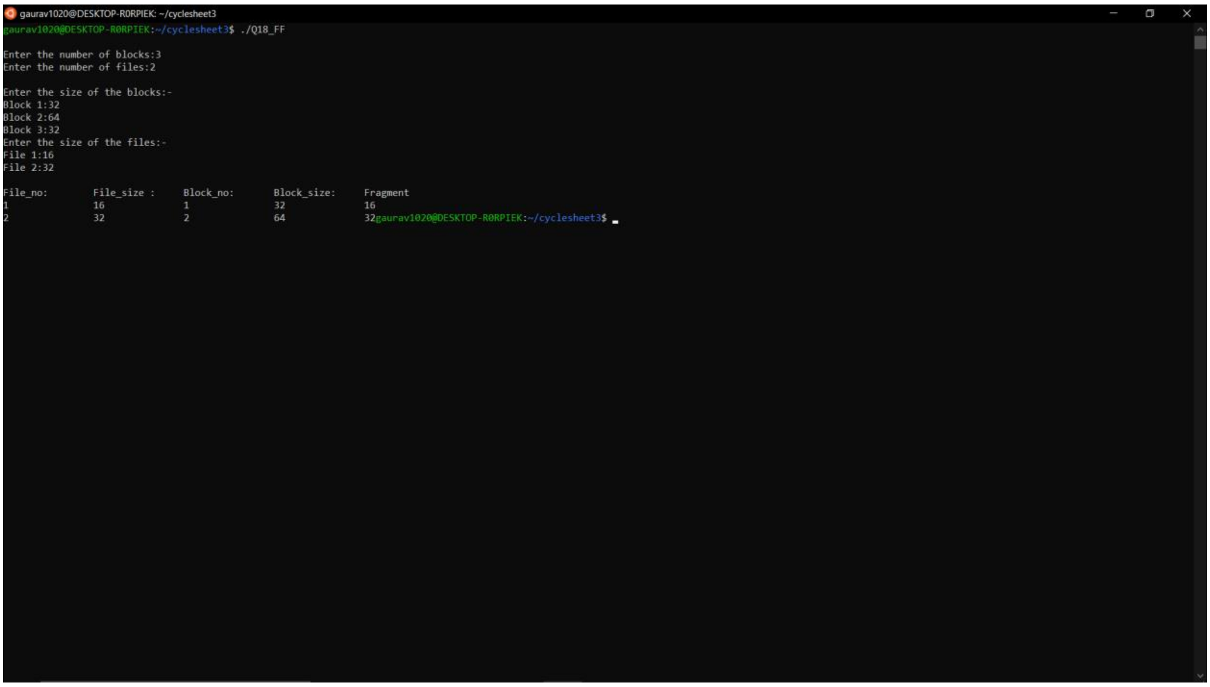
CODE

```
#include<stdio.h>

#include<unistd.h>

#define max 25

int main()

{

    int frag[max],b[max],f[max],i,j,nb,nf,temp;

    static int bf[max],ff[max];

    printf("\nEnter the number of blocks:");

    scanf("%d",&nb);

    printf("Enter the number of files:");

    scanf("%d",&nf);

    printf("\nEnter the size of the blocks:-\n");

    for(i=1;i<=nb;i++)

    {

        printf("Block %d:",i);

        scanf("%d",&b[i]);
    }
    printf("Enter the size of the files:-\n");
    for(i=1;i<=nf;i++)
    {
        printf("File %d:",i);
        scanf("%d",&f[i]);
    }
    for(i=1;i<=nf;i++)
    {
        for(j=1;j<=nb;j++)
        {
            if(bf[j]!=1)
            {
                temp=b[j]-f[i];
                if(temp>=0)
                {
                    ff[i]=j;
                    break;
                }
            }
        }
        frag[i]=temp;
        bf[ff[i]]=1;
    }
    printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragment");
    for(i=1;i<=nf;i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
    return 0;
}
```

OUTPUT



```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3
#include<stdio.h>
#include<unistd.h>
#define max 25
int main()
{
        int frag[max],b[max],f[max],i,j,nb,nf,temp;
        static int bf[max],ff[max];
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
        for(i=1;i<=nb;i++)
        {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
        }
        printf("Enter the size of the files:-\n");
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1)
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                {
                                        ff[i]=j;
                                        break;
                                }
                        }
                }
                frag[i]=temp;
                bf[ff[i]]=1;
        }
        printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragment");
        for(i=1;i<=nf;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        return 0;
}
"Q18_FF.c" 45L, 851C                                                                                45,1          All
```



```
gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ ./Q18_FF

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:32
Block 2:64
Block 3:32
Enter the size of the files:-
File 1:16
File 2:32

File_no:        File_size :     Block_no:       Block_size:     Fragment
1               16              1               32              16
2               32              2               64              32gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$
```

b. BEST FIT

CODE

#include<stdio.h>

#include<unistd.h>

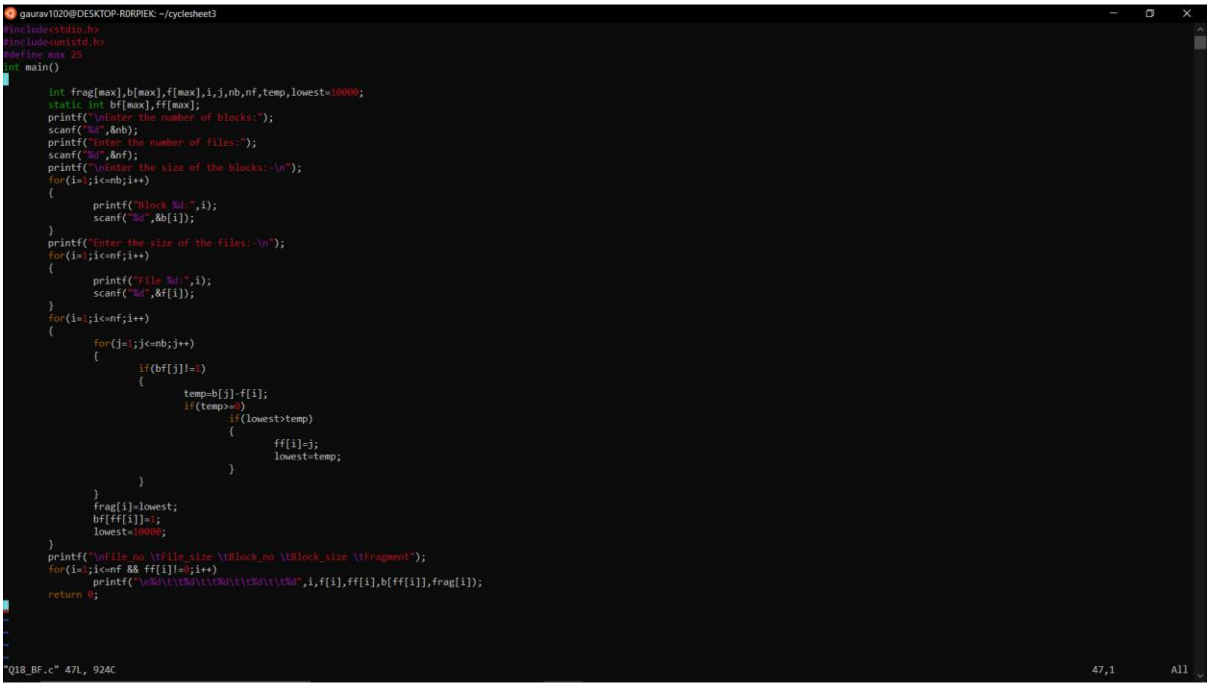
#define max 25

int main()

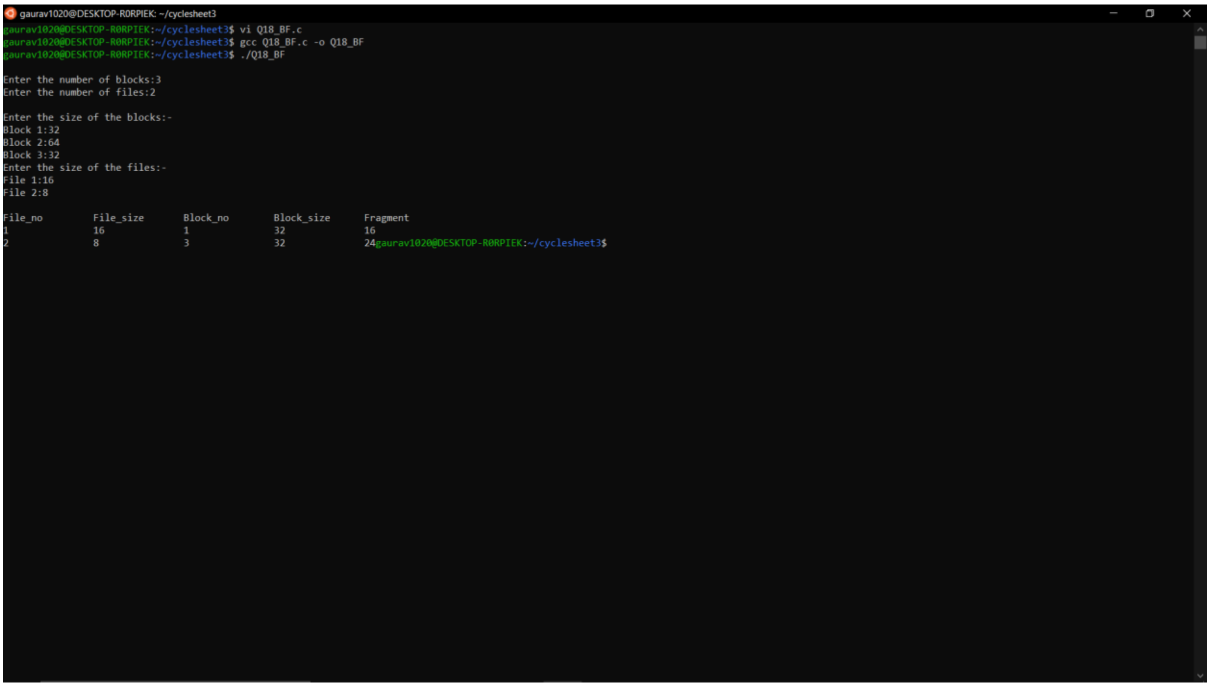
{

```
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;

static int bf[max],ff[max];

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of files:");

scanf("%d",&nf);

printf("\nEnter the size of the blocks:-\n");

for(i=1;i<=nb;i++)

{

        printf("Block %d:",i);

        scanf("%d",&b[i]);

}

printf("Enter the size of the files:-\n");

for(i=1;i<=nf;i++)

{

        printf("File %d:",i);

        scanf("%d",&f[i]);

}

for(i=1;i<=nf;i++)

{

        for(j=1;j<=nb;j++)

        {

                if(bf[j]!=1)

                {

                        temp=b[j]-f[i];

                        if(temp>=0)

                                if(lowest>temp)

                                {

                                        ff[i]=j;

                                        lowest=temp;

                                }
                }
            }
            frag[i]=lowest;
            bf[ff[i]]=1;
            lowest=10000;
        }
        printf("\nFile_no \tFile_size \tBlock_no \tBlock_size \tFragment");
        for(i=1;i<=nf && ff[i]!=0;i++)
            printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        return 0;
}
```

OUTPUT



```
#include<stdio.h>
#include<unistd.h>
#define max 25
int main()

        int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
        static int bf[max],ff[max];
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
        for(i=1;i<=nb;i++)
        {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
        }
        printf("Enter the size of the files:-\n");
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1)
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                        if(lowest>temp)
                                        {
                                                ff[i]=j;
                                                lowest=temp;
                                        }
                        }
                }
                frag[i]=lowest;
                bf[ff[i]]=1;
                lowest=10000;
        }
        printf("\nFile_no \tFile_size \tBlock_no \tBlock_size \tFragment");
        for(i=1;i<=nf && ff[i]!=0;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        return 0;

~
~
~
~
"Q18_BF.c" 47L, 924C                                                        47,1          All
```



gaurav1020@DESKTOP-R0RPIEK: ~/cyclesheet3

```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ vi Q18_BF.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ gcc Q18_BF.c -o Q18_BF
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ ./Q18_BF

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:32
Block 2:64
Block 3:32
Enter the size of the files:-
File 1:16
File 2:8

File_no         File_size       Block_no        Block_size      Fragment
1               16              1               32              16
2               8               3               32              24gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$
```

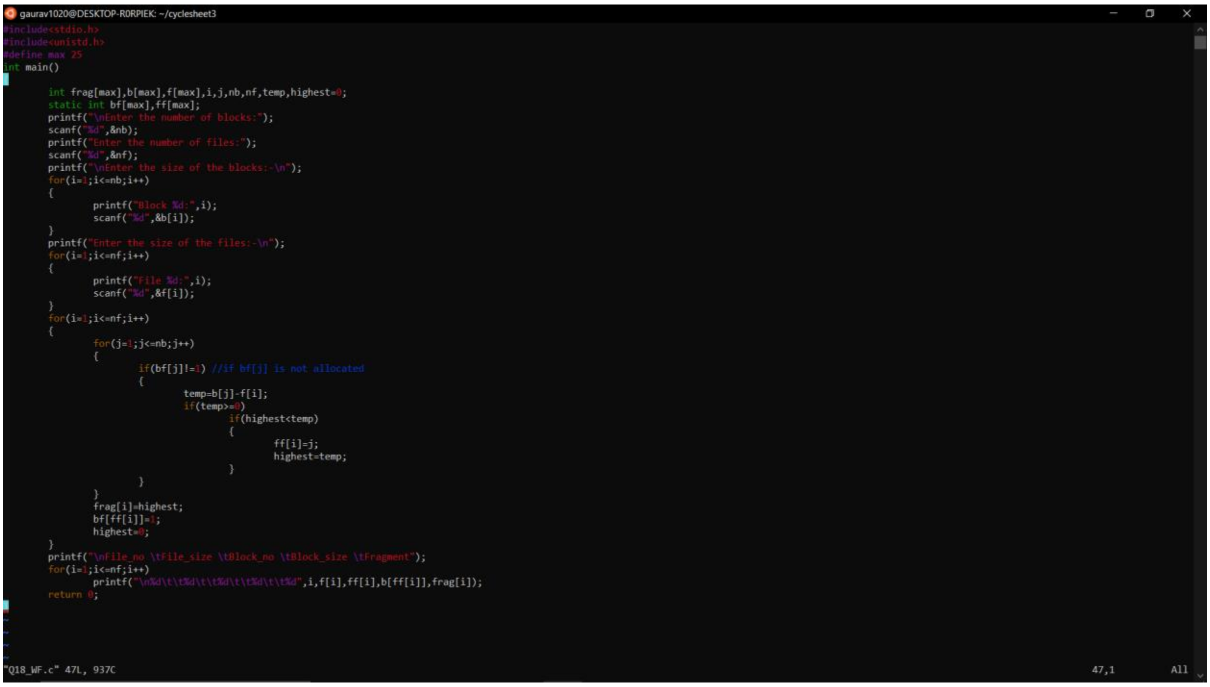
c. WORST FIT

CODE

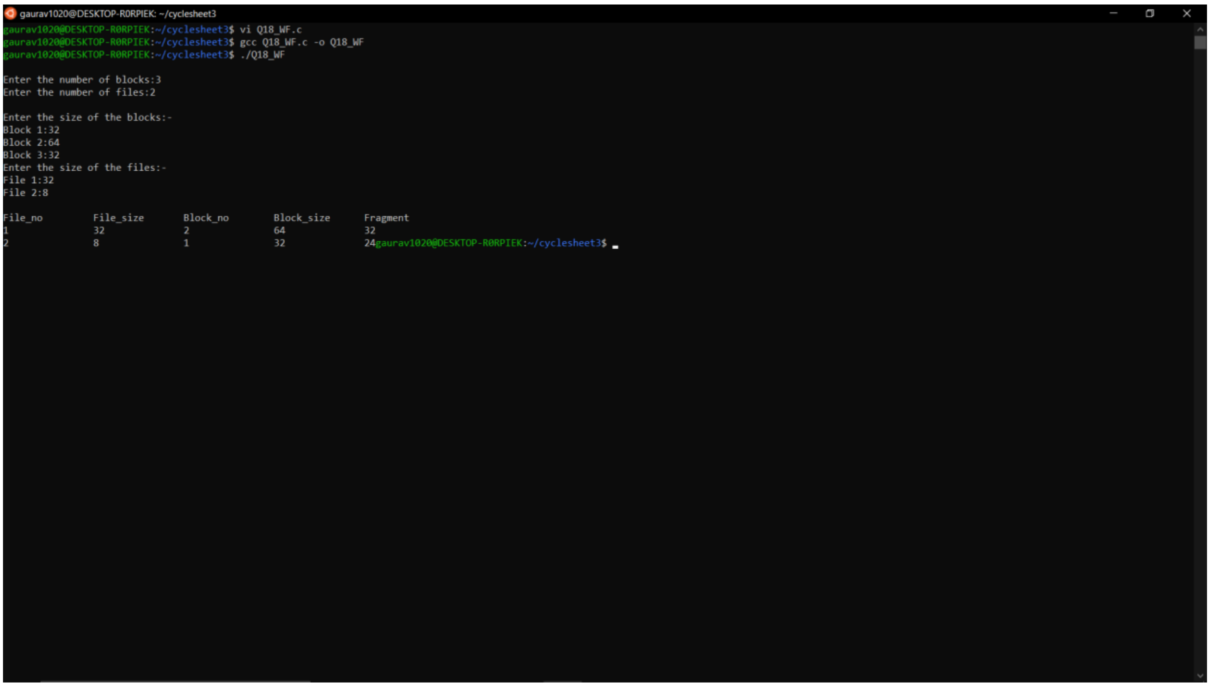
```
#include<stdio.h>

#include<unistd.h>

#define max 25

int main()

{

	int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;

	static int bf[max],ff[max];

	printf("\nEnter the number of blocks:");

	scanf("%d",&nb);

	printf("Enter the number of files:");

	scanf("%d",&nf);

	printf("\nEnter the size of the blocks:-\n");

	for(i=1;i<=nb;i++)

	{

		printf("Block %d:",i);

		scanf("%d",&b[i]);

	}
```

```
    printf("Enter the size of the files:-\n");
    for(i=1;i<=nf;i++)
    {
        printf("File %d:",i);
        scanf("%d",&f[i]);
    }
    for(i=1;i<=nf;i++)
    {
        for(j=1;j<=nb;j++)
        {
            if(bf[j]!=1) //if bf[j] is not allocated
            {
                temp=b[j]-f[i];
                if(temp>=0)
                    if(highest<temp)
                    {
                        ff[i]=j;
                        highest=temp;
                    }
            }
        }
        frag[i]=highest;
        bf[ff[i]]=1;
        highest=0;
    }
    printf("\nFile_no \tFile_size \tBlock_no \tBlock_size \tFragment");
    for(i=1;i<=nf;i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
    return 0;
}
```

OUTPUT



```
#include<stdio.h>
#include<unistd.h>
#define max 25
int main()
{

        int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
        static int bf[max],ff[max];
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
        for(i=1;i<=nb;i++)
        {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
        }
        printf("Enter the size of the files:-\n");
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1) //if bf[j] is not allocated
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                        if(highest<temp)
                                        {
                                                ff[i]=j;
                                                highest=temp;
                                        }
                        }
                }
                frag[i]=highest;
                bf[ff[i]]=1;
                highest=0;
        }
        printf("\nFile_no \tFile_size \tBlock_no \tBlock_size \tFragment");
        for(i=1;i<=nf;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        return 0;
}
~
~
~
~
"Q18_WF.c" 47L, 937C                                                                                    47,1          All
```



```
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ vi Q18_WF.c
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ gcc Q18_WF.c -o Q18_WF
gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ ./Q18_WF

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:32
Block 2:64
Block 3:32
Enter the size of the files:-
File 1:32
File 2:8

File_no         File_size       Block_no        Block_size      Fragment
1               32              2               64              32
2               8               1               32              24gaurav1020@DESKTOP-R0RPIEK:~/cyclesheet3$ _
```