Operating Systems Lab

1(a)

- To create file, writing data, making changes:
  - vi file_name
  - type “i” to enter “insert” mode and then write some data into the file.
  - now type “esc”(escape key) to exit the insert mode.
  - now type “:wq” to save and exit from the file.
  - check the changes you’ve made to the file by either opening the file again in vi editor or using the “cat file_name” command.

1(b)

- Different Unix commands to apply to the file created in the 1(a):
  - “cat file_name” concatenate the contents of the file to the terminal.
  - echo “Hello this is another line” >> file_name
    - this command appends the string given in quotes to the file.
  - ls
    - this command lists the files in the current directory.
  - cp file_name new_file_name
    - this command copies the contents of the file “file_name” to a new file “new_file_name”.
    - this command can also be used to copy the file from current directory to another directory:
      - cp file1 subdir/file1_copy
    - this command copies file “file1” into the subdirectory “subdir” and names the file “file1_copy”.
  - mv file_name new_file_name
    - this command is similar to that of “cp” but it deletes the old file “file_name” and only the “new_file_name” exists with all the same contents of “file_name”.
    - it can also be used to move files from one location to another:
      - mv file_name subdir/file_name_moved

2

Case statement

- Case statement script:

```
#!/bin/bash

FRUIT="kiwi"

case "$FRUIT" in
"apple") echo "Apple pie is quite tasty."
;;
"banana") echo "I like banana nut bread."
;;
"kiwi") echo "New Zealand is famous for kiwi."
;;
esac
```

- Case output:
  -
    ```
    [claw@wolf OS]$ bash shell_case
    New Zealand is famous for kiwi.
    ```

2

If statement

- If statement script:

```
#!/bin/bash

a=10
b=20

if [ $a == $b ]
then
echo "a is equal to b"
else
echo "a is not equal to b"
fi
```

- If output:
  -
    ```
    a is not equal to b
    ```

---

3 [do while]

- Shell do while:

```
#!/bin/bash

#i=16
while
         echo "this command is executed at least once $i"
        : ${start=$i}              # capture the starting value of i
        # some other commands      # needed for the loop
        (( ++i < 20 ))             # Place the loop ending test here.
do :; done
echo "Final value of $i///$start"
echo "The loop was executed $(( i - start )) times "
```

- dowhile output:

```
this command is executed at least once
this command is executed at least once 1
this command is executed at least once 2
this command is executed at least once 3
this command is executed at least once 4
this command is executed at least once 5
this command is executed at least once 6
this command is executed at least once 7
this command is executed at least once 8
this command is executed at least once 9
this command is executed at least once 10
this command is executed at least once 11
this command is executed at least once 12
this command is executed at least once 13
this command is executed at least once 14
this command is executed at least once 15
this command is executed at least once 16
this command is executed at least once 17
this command is executed at least once 18
this command is executed at least once 19
Final value of 20///
The loop was executed 20 times
```

---

3 [while]

- Shell while:

```
#!/bin/bash

a=0
while [ "$a" -lt 10 ]    # this is loop1
do
   b="$a"
   while [ "$b" -ge 0 ]  # this is loop2
   do
      echo -n "$b "
      b=`expr $b - 1`
   done
   echo
   a=`expr $a + 1`
done
```

- shell output:

```
0
1 0
2 1 0
3 2 1 0
4 3 2 1 0
5 4 3 2 1 0
6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
8 7 6 5 4 3 2 1 0
9 8 7 6 5 4 3 2 1 0
```

---

3 [for loop]

- Shell for loop:

```
#!/bin/bash

for i in {2..10}
do
    echo "output: $i"
done
```

- for loop output:

```
output: 2
output: 3
output: 4
output: 5
output: 6
output: 7
output: 8
output: 9
output: 10
```

4(a)

- Script:

```
#!/bin/bash
echo "Input number"
```

```
read no
echo "Input power"
read power

counter=0
ans=1

while [ $power -ne $counter ]
do
        ans=`expr $ans \* $no`
        counter=`expr $counter + 1`
done

echo "$no power of $power is $ans"
```

- output:
  - ```
    Input number
    3
    Input power
    2
    3 power of 2 is 9
    ```

4(b)

- Script:

```
#!/bin/bash
echo "Enter the file name: "
read file
if [ -f $file ]
then
echo "$file" " ---> It is a ORDINARY FILE."
elif [ -d $file ]
then
echo "$file" " ---> It is a DIRCTORY."
else
echo "$file" " ---> It is something else."
fi
```

- output:
  - ```
    Enter the file name:
    test.txt
    test.txt  ---> It is a ORDINARY FILE.
    ```

5(a)

- Script:

```
#!/bin/bash

echo "enter the filename"

read fname

echo "enter the starting line number"

read s

echo "enter the ending line number"

read n

sed -n $s,$n\p $fname | cat > newline

cat newline
```

- output:
  - ```
    enter the filename
    test
    enter the starting line number
    1
    enter the ending line number
    2
    abc
    hello
    ```

5(b)

- Script:

```
#!/bin/bash

echo "enter a file name"
read fname
echo "enter a word"
read s

sed -ie '/'"$s"'/d' "$fname"
```

- output:
```

```
[claw@wolf OS]$ cat test
hello
123
[claw@wolf OS]$ bash five_b
enter a file name
test
enter a word
hello
[claw@wolf OS]$ cat test
123
```

6(a)

- Script:

```
#!/bin/bash
echo -n "Enter file name : "
read file

# find out if file has write permission or not
[ -w $file ] && W="Write = yes" || W="Write = No"

# find out if file has excute permission or not
[ -x $file ] && X="Execute = yes" || X="Execute = No"

# find out if file has read permission or not
[ -r $file ] && R="Read = yes" || R="Read = No"

echo "$file permissions"
echo "$W"
echo "$R"
echo "$X"
```

- output:

```
Enter file name : test
test permissions
Write = yes
Read = yes
Execute = No
```

6(b)

- Script:

```
#!/bin/bash

echo "enter a file name: "
read fname

echo "enter a word: "
read fword

echo "Number of occurances is: "
grep -cow "$fword" "$fname"
```

- output:

```
[claw@wolf OS]$ bash six_b
enter a file name:
test
enter a word:
abc
Number of occurances is:
2
[claw@wolf OS]$ cat test
123
abc
abc
```

7

- Script:

```
#!/bin/python

import os
import shutil

#The following function lists the files and directories in the present working directory:
print(os.listdir())

#The following function moves moves the file from one destination to another:
os.mkdir('testdir')
os.mknod('oldfile')
os.rename('oldfile', 'testdir/oldfile')

#The following function will make a copy of a file:
os.mknod('filename')
shutil.copy('filename', 'filename_copy')
```

- output:

- ```
  [claw@wolf ulul]$ python unix.py
  ['unix.py', 'test.txt']
  ```

8

- Script:

```
#import fileinput
import string

x = []
#for line in fileinput.input():
#   x.append(line)

with open('eight_ascii', 'r') as f:
    for i in f.readline():
        x.append(i)

print(x)

y = []
for i in x:
    if i in string.ascii_lowercase:
        y.append(str.upper(i))

y = ''.join(y)
print(y)
```

- ASCII file:

```
abc()&^%$#@!&
```

- output:
  - ```
    [claw@wolf OS]$ python eight_prog.py
    ['a', 'b', 'c', '(', ')', '&', '^', '%', '$', '#', '@', '!', '&', '*', '\n']
    ABC
    ```

9

- Script:

```
import re

with open('test', 'r') as f:
    for line in f:
        pattern = re.compile(r'abc')
        matches = pattern.finditer(line)
        for match in matches:
            print(match.string)
```

- file being searched:

```
123
abc
abc
```

- output:
  - ```
    [claw@wolf OS]$ cat test
    123
    abc
    abc
    [claw@wolf OS]$ python pattern.py
    abc

    abc
    ```

---

10 [FCFS]

- Program:

```
#include<iostream>

using namespace std;

int main()
{
    int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
    cout<<"Enter total number of processes(maximum 20):";
    cin>>n;

    cout<<"\nEnter Process Burst Time\n";
    for(i=0;i<n;i++)
    {
        cout<<"P["<<i+1<<"]:";
        cin>>bt[i];
    }

    wt[0]=0;    //waiting time for first process is 0

    //calculating waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    }
```
```

```
    cout<<"\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time";

    //calculating turnaround time
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
        cout<<"\nP["<<i+1<<"]"<<"\t\t"<<bt[i]<<"\t\t"<<wt[i]<<"\t\t"<<tat[i];
    }

    avwt/=i;
    avtat/=i;
    cout<<"\n\nAverage Waiting Time:"<<avwt;
    cout<<"\nAverage Turnaround Time:"<<avtat;

    return 0;
}
```

- output:

```
[claw@wolf OS]$ g++ fcfs.cpp -o fcfs
[claw@wolf OS]$ ./fcfs
Enter total number of processes(maximum 20):3

Enter Process Burst Time
P[1]:2
P[2]:3
P[3]:5

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            2               0               2
P[2]            3               2               5
P[3]            5               5               10

Average Waiting Time:2
Average Turnaround Time:5[claw@wolf OS]$
```

---

11 [SJF]

- Program:

```
#include<stdio.h>

void main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time:\n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;           //contains process number
    }

    //sorting burst time in ascending order using selection sort
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
        }

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;            //waiting time for first process will be zero

    //calculate waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    total+=wt[i];
    }

    avg_wt=(float)total/n;      //average waiting time
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        total+=tat[i];
        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;     //average turnaround time
```

```
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

- output:

```
[claw@wolf OS]$ gcc sjf.c -o sjf
[claw@wolf OS]$ ./sjf
Enter number of process:3

Enter Burst Time:
p1:2
p2:3
p3:4

Process     Burst Time          Waiting Time    Turnaround Time
p1                 2                 0                   2
p2                 3                 2                   5
p3                 4                 5                   9

Average Waiting Time=2.333333
Average Turnaround Time=5.333333
```

---

12 [Priority based schelduling]

- Program:

```
#include<iostream>

using namespace std;

int main()
{
    int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
    cout<<"Enter Total Number of Process:";
    cin>>n;

    cout<<"\nEnter Burst Time and Priority\n";
    for(i=0;i<n;i++)
    {
        cout<<"\nP["<<i+1<<"]\n";
        cout<<"Burst Time:";
        cin>>bt[i];
        cout<<"Priority:";
        cin>>pr[i];
        p[i]=i+1;           //contains process number
    }

    //sorting burst time, priority and process number in ascending order using selection sort
        for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(pr[j]<pr[pos])
                pos=j;
        }

        temp=pr[i];
        pr[i]=pr[pos];
        pr[pos]=temp;

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;            //waiting time for first process is zero

    //calculate waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=total/n;      //average waiting time
    total=0;

    cout<<"\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time";
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        total+=tat[i];
        cout<<"\nP["<<p[i]<<"]\t\t  "<<bt[i]<<"\t\t    "<<wt[i]<<"\t\t\t"<<tat[i];
    }

    avg_tat=total/n;     //average turnaround time
    cout<<"\n\nAverage Waiting Time="<<avg_wt;
    cout<<"\nAverage Turnaround Time="<<avg_tat;
  return 0;
}
```

- output:

```
[claw@wolf OS]$ g++ priority.cpp -o priority
[claw@wolf OS]$ ./priority
Enter Total Number of Process:3

Enter Burst Time and Priority

P[1]
Burst Time:1
Priority:3

P[2]
Burst Time:2
Priority:2

P[3]
Burst Time:3
Priority:1

Process     Burst Time          Waiting Time    Turnaround Time
P[3]              3                 0                   3
P[2]              2                 3                   5
P[1]              1                 5                   6

Average Waiting Time=2
Average Turnaround Time=4[claw@wolf OS]$
```

---

13 [Round Robin]

- Program:

```
#include<stdio.h>

int main()
{

  int count,j,n,time,remain,flag=0,time_quantum;
  int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
  printf("Enter Total Process:\t ");
  scanf("%d",&n);
  remain=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
    scanf("%d",&at[count]);
    scanf("%d",&bt[count]);
    rt[count]=bt[count];
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
      rt[count]=0;
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
      wait_time+=time-at[count]-bt[count];
      turnaround_time+=time-at[count];
      flag=0;
    }
    if(count==n-1)
    count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}
```

- output:

```
[claw@wolf OS]$ gcc roundrobin.c -o roundrobin
[claw@wolf OS]$ ./roundrobin
Enter Total Process:     3
Enter Arrival Time and Burst Time for Process Process Number 1 :0
3
Enter Arrival Time and Burst Time for Process Process Number 2 :0
4
Enter Arrival Time and Burst Time for Process Process Number 3 :0
3
Enter Time Quantum:     1


Process |Turnaround Time|Waiting Time

P[1]    |       7       |       4
P[3]    |       9       |       6
P[2]    |       10      |       6

Average Waiting Time= 5.333333
Avg Turnaround Time = 8.666667[claw@wolf OS]$
```

---

14 [Semaphores]

- Program:

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

int mutex=1,full=0,empty=3,x=0;

int main()
{
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n1.Producer\n2.Consumer\n3.Exit");
    while(1)
    {
        printf("\nEnter your choice:");
        scanf("%d",&n);
        switch(n)
        {
            case 1:    if((mutex==1)&&(empty!=0))
                        producer();
                    else
                        printf("Buffer is full!!");
                    break;
            case 2:    if((mutex==1)&&(full!=0))
                        consumer();
                    else
                        printf("Buffer is empty!!");
                    break;
            case 3:
                    exit(0);
                    break;
        }
    }
return 0;
}

int wait(int s)
{
    return (--s);
}

int signal(int s)
{
    return(++s);
}

void producer()
{
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
    printf("\nProducer produces the item %d",x);
    mutex=signal(mutex);
}

void consumer()
{
    mutex=wait(mutex);
    full=wait(full);
    empty=signal(empty);
    printf("\nConsumer consumes item %d",x);
    x--;
    mutex=signal(mutex);
}
```

- output:

```
[claw@wolf OS]$ gcc semaphore.c -o semaphore
[claw@wolf OS]$ ./semaphore

1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:2

Consumer consumes item 1
Enter your choice:3
```

---

15 [FIFO]

- Program:

```
#include<stdio.h>

int main()
{
      int reference_string[10], page_faults = 0, m, n, s, pages, frames;
      printf("\nEnter Total Number of Pages:\t");
      scanf("%d", &pages);
      printf("\nEnter values of Reference String:\n");
      for(m = 0; m < pages; m++)
      {
            printf("Value No. [%d]:\t", m + 1);
            scanf("%d", &reference_string[m]);
      }
      printf("\nEnter Total Number of Frames:\t");
      {
            scanf("%d", &frames);
      }
      int temp[frames];
      for(m = 0; m < frames; m++)
      {
            temp[m] = -1;
      }
      for(m = 0; m < pages; m++)
      {
            s = 0;
            for(n = 0; n < frames; n++)
            {
                  if(reference_string[m] == temp[n])
                  {
                        s++;
                        page_faults--;
                  }
            }
            page_faults++;
            if((page_faults <= frames) && (s == 0))
            {
                  temp[m] = reference_string[m];
            }
            else if(s == 0)
            {
                  temp[(page_faults - 1) % frames] = reference_string[m];
            }
            printf("\n");
            for(n = 0; n < frames; n++)
            {
                  printf("%d\t", temp[n]);
            }
      }
      printf("\nTotal Page Faults:\t%d\n", page_faults);
      return 0;
}
```

- output:

```
Enter Total Number of Pages:    8

Enter values of Reference String:
Value No. [1]:  1
Value No. [2]:  2
Value No. [3]:  3
Value No. [4]:  4
Value No. [5]:  2
Value No. [6]:  1
Value No. [7]:  5
Value No. [8]:  6

Enter Total Number of Frames:   4

1        -1       -1       -1
1        2        -1       -1
1        2        3        -1
1        2        3        4
1        2        3        4
1        2        3        4
5        2        3        4
5        6        3        4
Total Page Faults:       6
```

---

15 [LRU]

- Program:

```
#include<stdio.h>

int findLRU(int time[], int n){
    int i, minimum = time[0], pos = 0;

    for(i = 1; i < n; ++i){
        if(time[i] < minimum){
            minimum = time[i];
            pos = i;
        }
    }

    return pos;
}

int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;
    printf("Enter number of frames: ");
    scanf("%d", &no_of_frames);

    printf("Enter number of pages: ");
    scanf("%d", &no_of_pages);

    printf("Enter reference string: ");

    for(i = 0; i < no_of_pages; ++i){
        scanf("%d", &pages[i]);
    }

    for(i = 0; i < no_of_frames; ++i){
        frames[i] = -1;
    }

    for(i = 0; i < no_of_pages; ++i){
        flag1 = flag2 = 0;

        for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == pages[i]){
                counter++;
                time[j] = counter;
                   flag1 = flag2 = 1;
                   break;
               }
        }

        if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
                if(frames[j] == -1){
                    counter++;
                    faults++;
                    frames[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    break;
                }
            }
        }

        if(flag2 == 0){
            pos = findLRU(time, no_of_frames);
            counter++;
            faults++;
            frames[pos] = pages[i];
            time[pos] = counter;
        }

        printf("\n");
```

```
        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }

    printf("\n\nTotal Page Faults = %d\n", faults);

    return 0;
}
```

- output:

```
Enter number of frames: 4
Enter number of pages: 8
Enter reference string: 1
3
4
5
2
6
7
1

1       -1      -1      -1
1       3       -1      -1
1       3       4       -1
1       3       4       5
2       3       4       5
2       6       4       5
2       6       7       5
2       6       7       1

Total Page Faults = 8
```

---

15 [LFU]

- Program:

```
#include<stdio.h>

int main()
{
      int total_frames, total_pages, hit = 0;
      int pages[25], frame[10], arr[25], time[25];
      int m, n, page, flag, k, minimum_time, temp;
      printf("Enter Total Number of Pages:\t");
      scanf("%d", &total_pages);
      printf("Enter Total Number of Frames:\t");
      scanf("%d", &total_frames);
      for(m = 0; m < total_frames; m++)
      {
            frame[m] = -1;
      }
      for(m = 0; m < 25; m++)
      {
            arr[m] = 0;
      }
      printf("Enter Values of Reference String\n");
      for(m = 0; m < total_pages; m++)
      {
            printf("Enter Value No.[%d]:\t", m + 1);
            scanf("%d", &pages[m]);
      }
      printf("\n");
      for(m = 0; m < total_pages; m++)
      {
            arr[pages[m]]++;
            time[pages[m]] = m;
            flag = 1;
            k = frame[0];
            for(n = 0; n < total_frames; n++)
            {
                  if(frame[n] == -1 || frame[n] == pages[m])
                  {
                        if(frame[n] != -1)
                        {
                              hit++;
                        }
                        flag = 0;
                        frame[n] = pages[m];
                        break;
                  }
                  if(arr[k] > arr[frame[n]])
                  {
                        k = frame[n];
                  }
            }
            if(flag)
            {
                  minimum_time = 25;
                  for(n = 0; n < total_frames; n++)
                  {
                        if(arr[frame[n]] == arr[k] && time[frame[n]] < minimum_time)
                        {
                              temp = n;
                              minimum_time = time[frame[n]];
                        }
                  }
                  arr[frame[temp]] = 0;
                  frame[temp] = pages[m];
            }
```

```
            for(n = 0; n < total_frames; n++)
            {
                  printf("%d\t", frame[n]);
            }
            printf("\n");
      }
      printf("Page Hit:\t%d\n", hit);
      return 0;

}
```

- output:

```
Enter Total Number of Pages:    8
Enter Total Number of Frames:   4
Enter Values of Reference String
Enter Value No.[1]:     3
Enter Value No.[2]:     4
Enter Value No.[3]:     5
Enter Value No.[4]:     6
Enter Value No.[5]:     2
Enter Value No.[6]:     1
Enter Value No.[7]:     3
Enter Value No.[8]:     4

3       -1      -1      -1
3       4       -1      -1
3       4       5       -1
3       4       5       6
2       4       5       6
2       1       5       6
2       1       3       6
2       1       3       4
Page Hit:       0
```

---

16 [Sequential]

- Program:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
for(i=0;i<50;i++)
f[i]=0;
printf("Files Allocated are : \n");
x: count=0;
printf("Enter starting block and length of files: ");
scanf("%d%d", &st,&len);
for(k=st;k<(st+len);k++)
 if(f[k]==0)
 count++;
if(len==count)
{
 for(j=st;j<(st+len);j++)
 if(f[j]==0)
 {
 f[j]=1;
 printf("%d\t%d\n",j,f[j]);
 }
 if(j!=(st+len-1))
 printf(" The file is allocated to disk\n");
}
else
printf(" The file is not allocated \n");
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
 goto x;
else
 exit;
getc;
}
```

- output:

```
[claw@wolf OS]$ gcc sequential.c -o sequential
[claw@wolf OS]$ ./sequential
Files Allocated are :
Enter starting block and length of files: 17 4
17      1
18      1
19      1
20      1
 The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 21 3
21      1
22      1
23      1
 The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)
```

16 [Indexed]

- Program:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;
for(i=0;i<50;i++)
f[i]=0;
x:printf("Enter the index block: ");
scanf("%d",&ind);
if(f[ind]!=1)
{
 printf("Enter no of blocks needed and no of files for the index %d on the disk : \n",
ind);
 scanf("%d",&n);
}
else
{
 printf("%d index is already allocated \n",ind);
 goto x;
}
y: count=0;
for(i=0;i<n;i++)
{
 scanf("%d", &index[i]);
 if(f[index[i]]==0)
 count++;
}
if(count==n)
{
 for(j=0;j<n;j++)
 f[index[j]]=1;
 printf("Allocated\n");
 printf("File Indexed\n");
 for(k=0;k<n;k++)
 printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);
}
else
{
printf("File in the index is already allocated \n");
printf("Enter another file indexed");
goto y;
}
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
 goto x;
else
 exit(0);
getc;
}
```

- Output:
  -
    ```
    [claw@wolf OS]$ gcc indexed.c -o indexed
    [claw@wolf OS]$ ./indexed
    Enter the index block: 5
    Enter no of blocks needed and no of files for the index 5 on the disk :
    4
    1 2 3 4
    Allocated
    File Indexed
    5-------->1 : 1
    5-------->2 : 1
    5-------->3 : 1
    5-------->4 : 1
    Do you want to enter more file(Yes - 1/No - 0)
    ```

16 [Linked]

- Program:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int f[50], p,i, st, len, j, c, k, a;
for(i=0;i<50;i++)
f[i]=0;
printf("Enter how many blocks already allocated: ");
scanf("%d",&p);
printf("Enter blocks already allocated: ");
for(i=0;i<p;i++)
{
 scanf("%d",&a);
 f[a]=1;
}
x: printf("Enter index starting block and length: ");
scanf("%d%d", &st,&len);
k=len;
if(f[st]==0)
{
 for(j=st;j<(st+k);j++)
 {
 if(f[j]==0)
 {
 f[j]=1;
 printf("%d-------->%d\n",j,f[j]);
 }
 else
 {
```

```
 printf("%d Block is already allocated \n",j);
 k++;
 }
 }
}
else
printf("%d starting block is already allocated \n",st);
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
 goto x;
else
 exit(0);
 getc;
}
```

- Output:
  -
    ```
    [claw@wolf OS]$ gcc linked.c -o linked
    [claw@wolf OS]$ ./linked
    Enter how many blocks already allocated: 3
    Enter blocks already allocated: 1 3 5
    Enter index starting block and length: 2
    4
    2-------->1
    3 Block is already allocated
    4-------->1
    5 Block is already allocated
    6-------->1
    7-------->1
    Do you want to enter more file(Yes - 1/No - 0)
    ```