Date - 17/8/2020

Experiment No. 2

Simulation of File Allocation Strategies

Aim: To write a program to simulate the following file allocation strategies.
a) Sequential b) Indexed c) Linked

Theory: The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

1.Contiguous or Sequential Allocation
2.Indexed Allocation
3.Linked Allocation

The main idea behind these methods is to provide an efficient disk space utilization and fast access to the file blocks.

These methods have their own advantages and disadvantages as discussed below:

1. Sequential Allocation

In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file. The directory entry for a file with contiguous allocation contains

a) Address of starting block
b) Length of the allocated portion.

Advantages:

1) Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the ‘k’th block of the file which starts at block ‘a’ can easily be obtained as (‘a+k’).

2) This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

Disadvantages:

1) This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.

2) Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

2. Indexed Allocation

In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block.

Advantages:

1) This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.

2) It overcomes the problem of external fragmentation.

Disadvantages:

1) The pointer overhead for indexed allocation is greater than linked allocation.

2) For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization.

3. Linked Allocation

In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

Advantages:

1) This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.

2) This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

Disadvantages:

1) Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.

2) It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.

3) Pointers required in the linked allocation incur some extra overhead.

Algorithm:

Sequential Allocation

Step 1: Start.

Step 2: Get the number of memory partitions and their sizes.

Step 3: Get the number of processes and values of block size for each process.

Step 4: First fit algorithm searches all the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.

Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to the requesting process and allocates it.

Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.

Step 7: Analyses all the three memory management techniques and displays the best algorithm which utilizes the memory resources effectively and efficiently.

Step 8: Stop the program.

Indexed Allocation

```
Step 1:  Start.
Step 2:  Let n be the size of the buffer
Step 3:  Check if there are any producer
Step 4:  If yes check whether the buffer is full
Step 5:  If no the producer item is stored in the buffer
Step 6:  If the buffer is full the producer has to wait
Step 7:  Check there is any consumer.If yes check whether the buffer is empty
Step 8:  If no the consumer consumes them from the buffer
Step 9:  If the buffer is empty, the consumer has to wait.
Step 10: Repeat checking for the producer and consumer till required
Step 11: Stop the process.
```

Linked Allocation

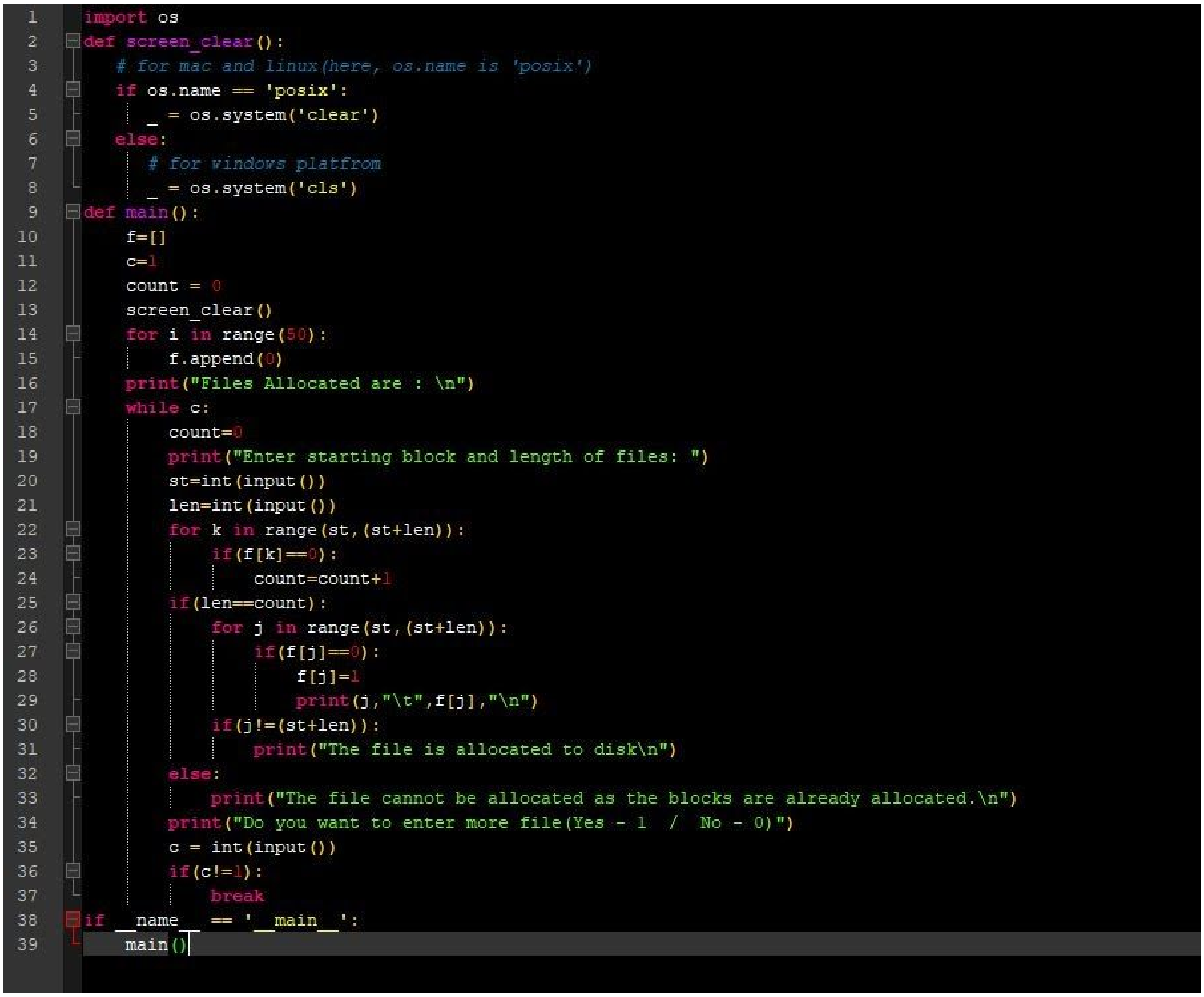
```
Step 1: Create a queue to hold all pages in memory
Step 2: When the page is required replace the page at the head of the queue
Step 3: Now the new page is inserted at the tail of the queue
Step 4: Create a stack
Step 5: When the page fault occurs replace page present at the bottom of the stack
Step 6: Stop the allocation.
```

Program:

<u>Sequential Allocation</u>

<u>Program</u>

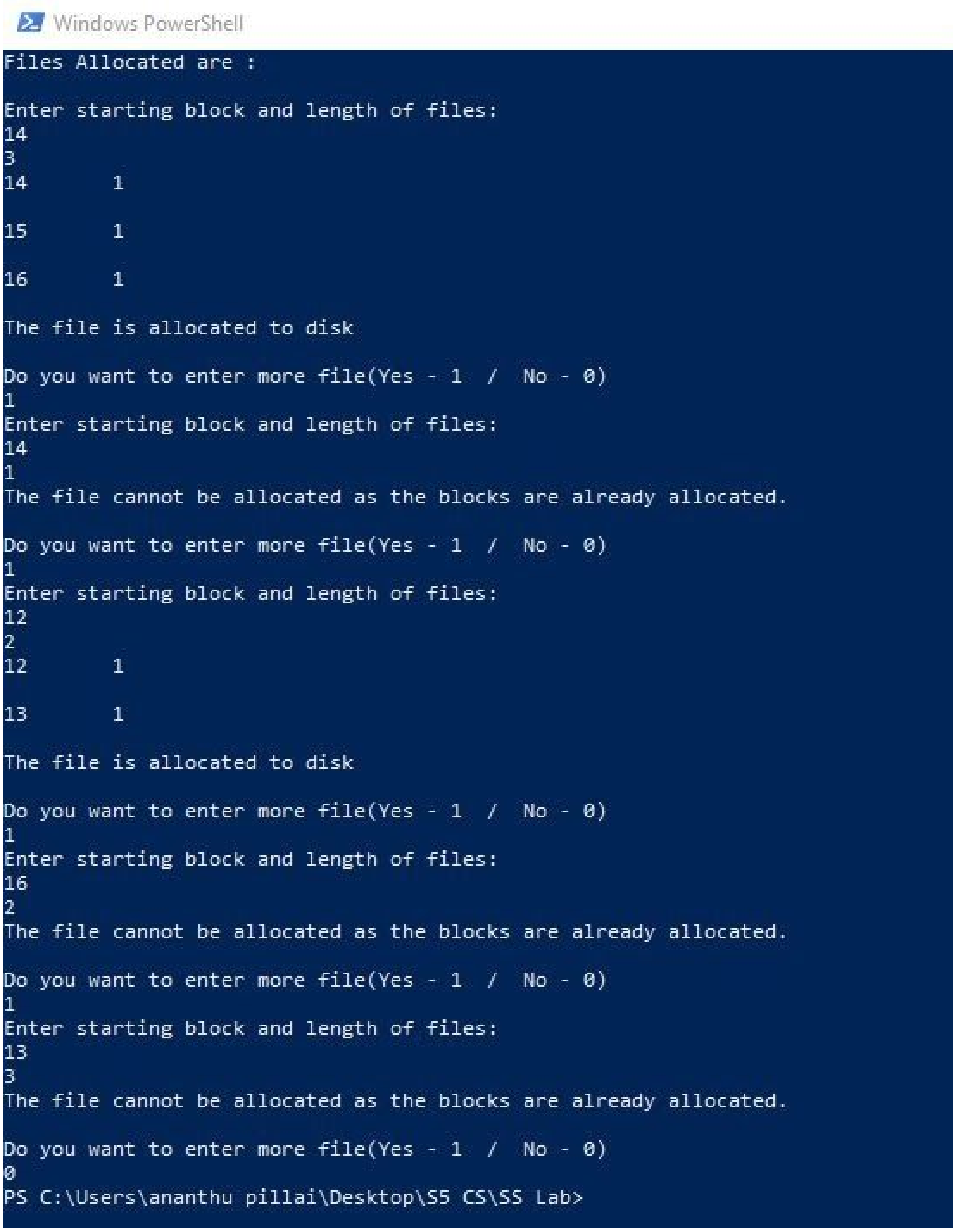
```
import os
def screen_clear():
  # for mac and linux(os.name is 'posix')
  if os.name == 'posix':
     _ = os.system('clear')
  else:
     # for windows platform
     _ = os.system('cls')
def main():
   f=[]
   c=1
   count = 0
   screen_clear()
   for i in range(50):
      f.append(0)
   print("Files Allocated are : \n")
   while c:
      count=0
      print("Enter starting block and length of files: ")
      st=int(input())
      len=int(input())
      for k in range(st,(st+len)):
         if(f[k]==0):
            count=count+1
      if(len==count):
            for j in range(st,(st+len)):
                if(f[j]==0):
                    f[j]=1
                    print(j,"\t",f[j],"\n")
            if(j!=(st+len)):
                print("The file is allocated to disk\n")
        else:
            print("The file cannot be allocated as the blocks are already allocated.\n")
        print("Do you want to enter more file(Yes - 1  /  No - 0)")
        c = int(input())
        if(c!=1):
            break
if __name__ == '__main__':
    main()
```



```
import os
def screen_clear():
   # for mac and linux(here, os.name is 'posix')
   if os.name == 'posix':
      _ = os.system('clear')
   else:
      # for windows platfrom
      _ = os.system('cls')
def main():
    f=[]
    c=1
    count = 0
    screen_clear()
    for i in range(50):
        f.append(0)
    print("Files Allocated are : \n")
    while c:
        count=0
        print("Enter starting block and length of files: ")
        st=int(input())
        len=int(input())
        for k in range(st,(st+len)):
            if(f[k]==0):
                count=count+1
        if(len==count):
            for j in range(st,(st+len)):
                if(f[j]==0):
                    f[j]=1
                    print(j,"\t",f[j],"\n")
            if(j!=(st+len)):
                print("The file is allocated to disk\n")
        else:
            print("The file cannot be allocated as the blocks are already allocated.\n")
        print("Do you want to enter more file(Yes - 1  /  No - 0)")
        c = int(input())
        if(c!=1):
            break
if __name__ == '__main__':
    main()
```

Output:

```
Files Allocated are :

Enter starting block and length of files:
14
3
14      1

15      1

16      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:
14
1
The file cannot be allocated as the blocks are already allocated.

Do you want to enter more file(Yes - 1  /  No - 0)
1
Enter starting block and length of files:
12
2
12      1

13      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:
16
2
The file cannot be allocated as the blocks are already allocated.

Do you want to enter more file(Yes - 1  /  No - 0)
1
Enter starting block and length of files:
13
3
The file cannot be allocated as the blocks are already allocated.

Do you want to enter more file(Yes - 1  /  No - 0)
0
```



Windows PowerShell

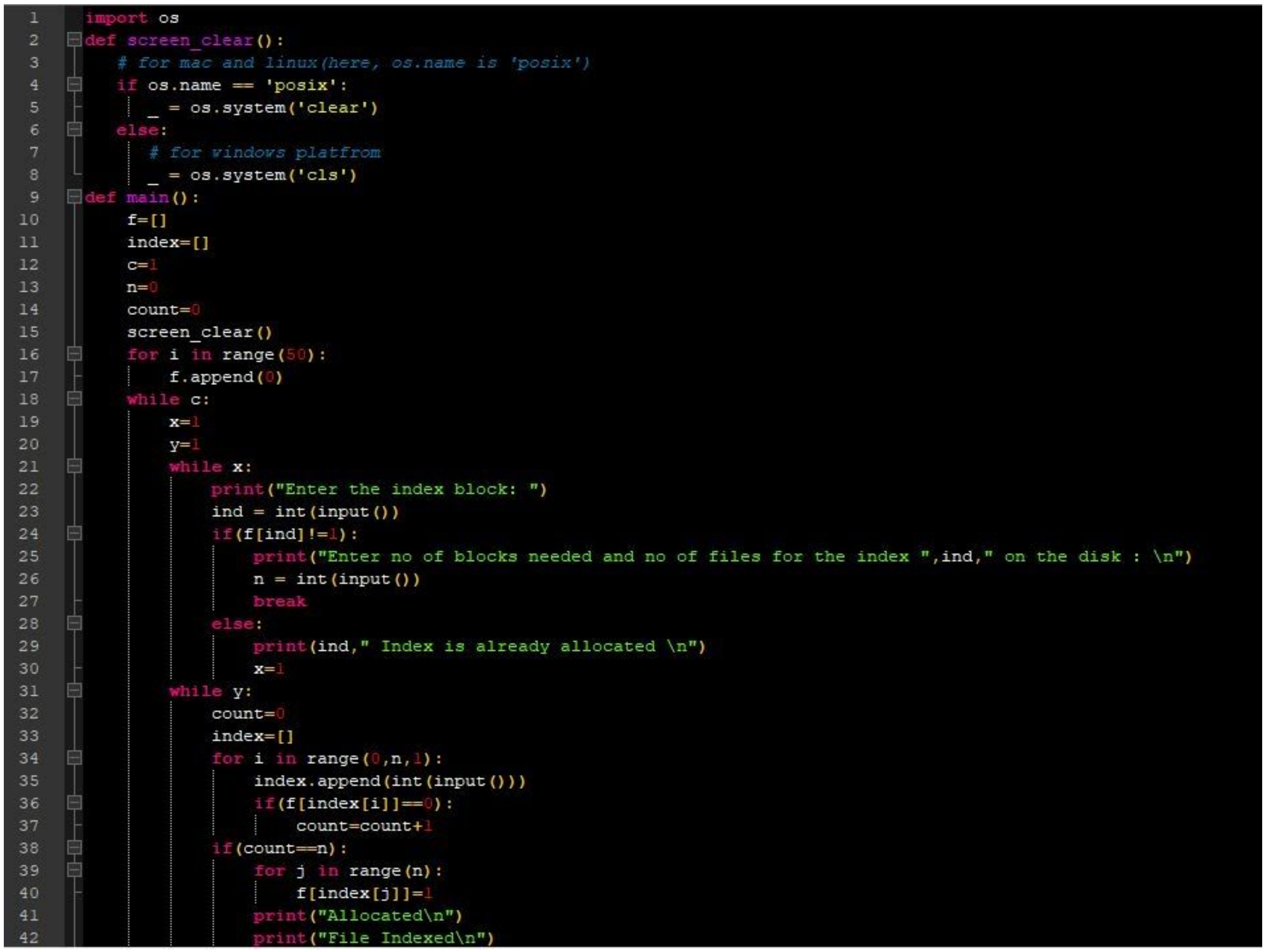
```
Files Allocated are :

Enter starting block and length of files:
14
3
14      1

15      1

16      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:
14
1
The file cannot be allocated as the blocks are already allocated.

Do you want to enter more file(Yes - 1  /  No - 0)
1
Enter starting block and length of files:
12
2
12      1

13      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:
16
2
The file cannot be allocated as the blocks are already allocated.

Do you want to enter more file(Yes - 1  /  No - 0)
1
Enter starting block and length of files:
13
3
The file cannot be allocated as the blocks are already allocated.

Do you want to enter more file(Yes - 1  /  No - 0)
0
PS C:\Users\ananthu pillai\Desktop\S5 CS\SS Lab>
```

Indexed Allocation

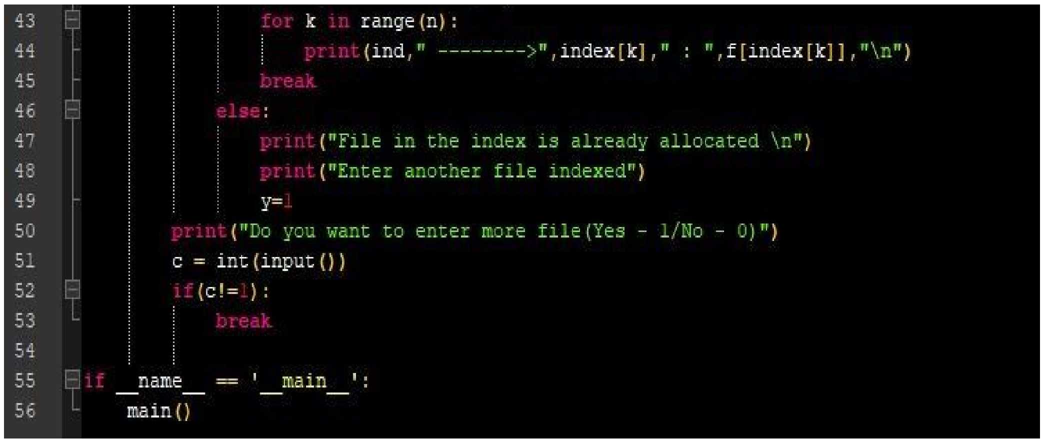
Program

```
import os
def screen_clear():
  # for mac and linux(os.name is 'posix')
  if os.name == 'posix':
    _ = os.system('clear')
  else:
    # for windows platform
    _ = os.system('cls')
def main():
   f=[]
   index=[]
   c=1
   n=0
   count=0
   screen_clear()
   for i in range(50):
      f.append(0)
   while c:
      x=1
      y=1
      while x:
         print("Enter the index block: ")
         ind = int(input())
         if(f[ind]!=1):
            print("Enter no of blocks needed and no of files for the index ",ind," on the disk : \n")
            n = int(input())
            break
         else:
            print(ind," Index is already allocated \n")
            x=1
      while y:
         count=0
         index=[]
         for i in range(0,n,1):
            index.append(int(input()))
            if(f[index[i]]==0):
               count=count+1
         if(count==n):
            for j in range(n):
               f[index[j]]=1
            print("Allocated\n")
            print("File Indexed\n")
            for k in range(n):
               print(ind," -------->",index[k]," : ",f[index[k]],"\n")
            break
         else:
            print("File in the index is already allocated \n")
            print("Enter another file indexed")
            y=1
```

```
        print("Do you want to enter more file(Yes - 1/No - 0)")
        c = int(input())
        if(c!=1):
            break

if __name__ == '__main__':
    main()
```



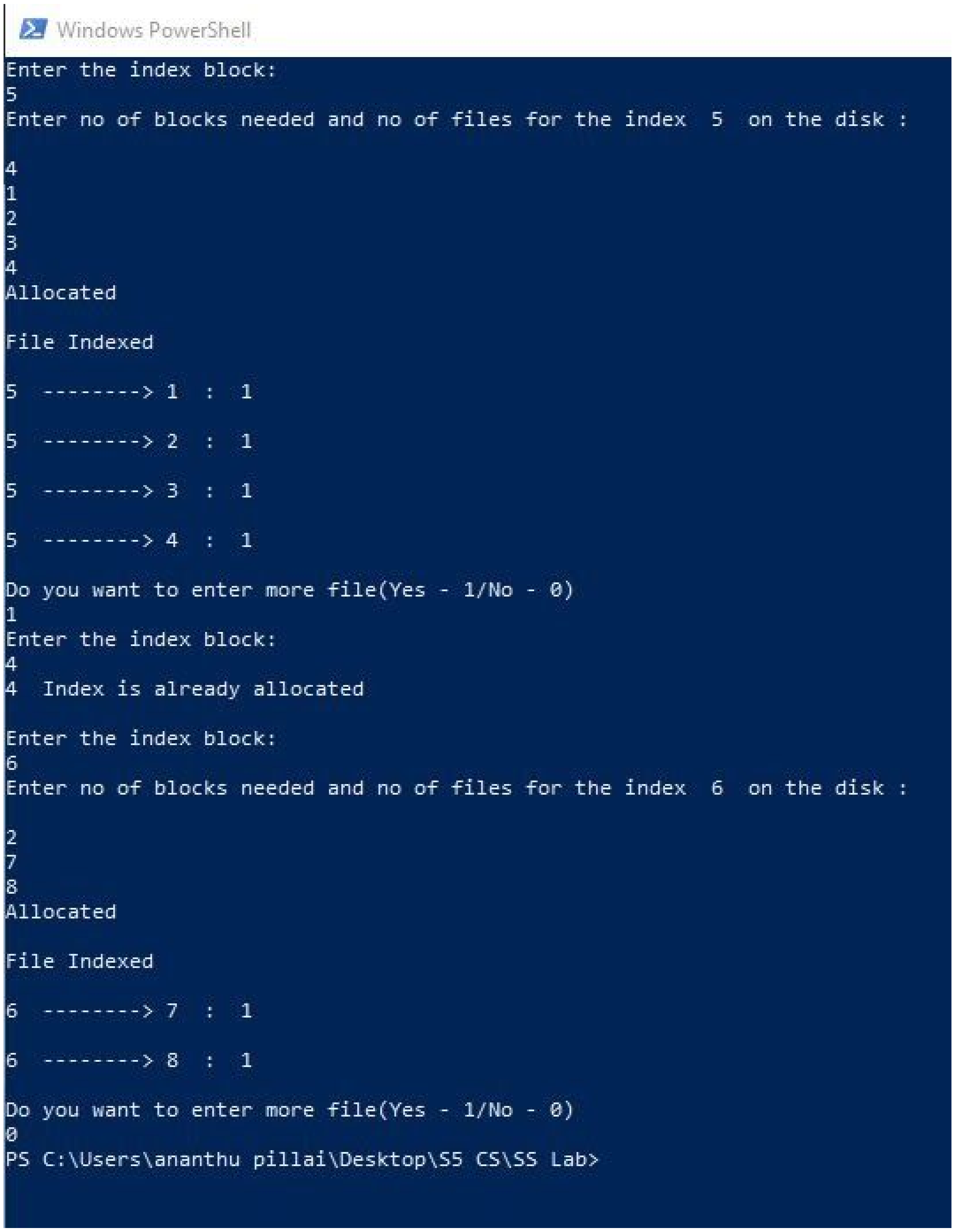
```
import os
def screen_clear():
    # for mac and linux(here, os.name is 'posix')
    if os.name == 'posix':
        _ = os.system('clear')
    else:
        # for windows platfrom
        _ = os.system('cls')
def main():
    f=[]
    index=[]
    c=1
    n=0
    count=0
    screen_clear()
    for i in range(50):
        f.append(0)
    while c:
        x=1
        y=1
        while x:
            print("Enter the index block: ")
            ind = int(input())
            if(f[ind]!=1):
                print("Enter no of blocks needed and no of files for the index ",ind," on the disk : \n")
                n = int(input())
                break
            else:
                print(ind," Index is already allocated \n")
                x=1
        while y:
            count=0
            index=[]
            for i in range(0,n,1):
                index.append(int(input()))
                if(f[index[i]]==0):
                    count=count+1
            if(count==n):
                for j in range(n):
                    f[index[j]]=1
                print("Allocated\n")
                print("File Indexed\n")
```



```
                for k in range(n):
                    print(ind," -------->",index[k]," : ",f[index[k]],"\n")
                break
            else:
                print("File in the index is already allocated \n")
                print("Enter another file indexed")
                y=1
        print("Do you want to enter more file(Yes - 1/No - 0)")
        c = int(input())
        if(c!=1):
            break

if __name__ == '__main__':
    main()
```

Output:

```
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)
1
Enter the index block:
4
4  Index is already allocated

Enter the index block:
6
Enter no of blocks needed and no of files for the index  6  on the disk :

2
7
8
Allocated

File Indexed

6  --------> 7  :  1

6  --------> 8  :  1

Do you want to enter more file(Yes - 1/No - 0)
0
```

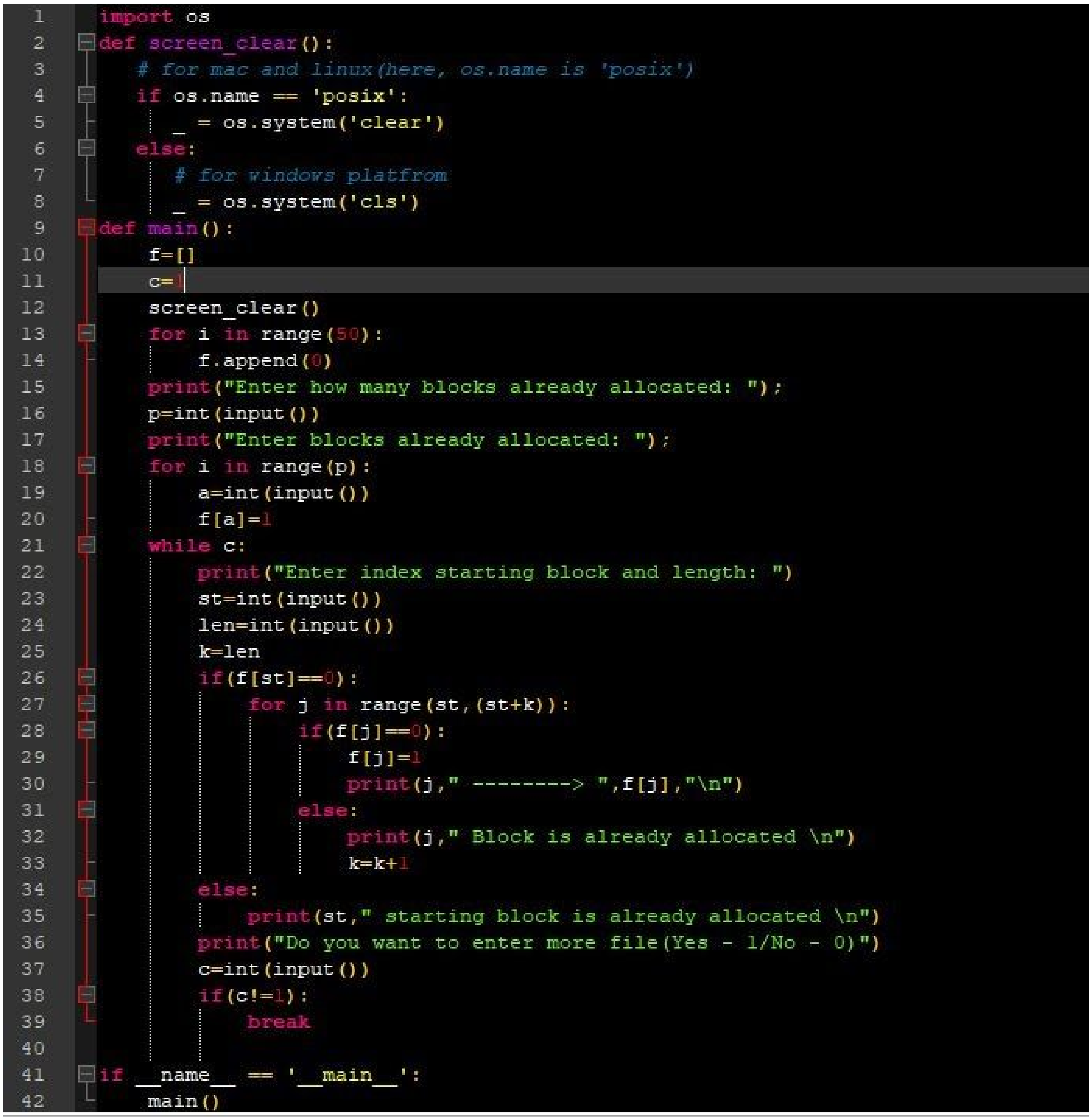

Windows PowerShell

```
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)
1
Enter the index block:
4
4  Index is already allocated

Enter the index block:
6
Enter no of blocks needed and no of files for the index  6  on the disk :

2
7
8
Allocated

File Indexed

6  --------> 7  :  1

6  --------> 8  :  1

Do you want to enter more file(Yes - 1/No - 0)
0
PS C:\Users\ananthu pillai\Desktop\S5 CS\SS Lab>
```

Linked Allocation

Program

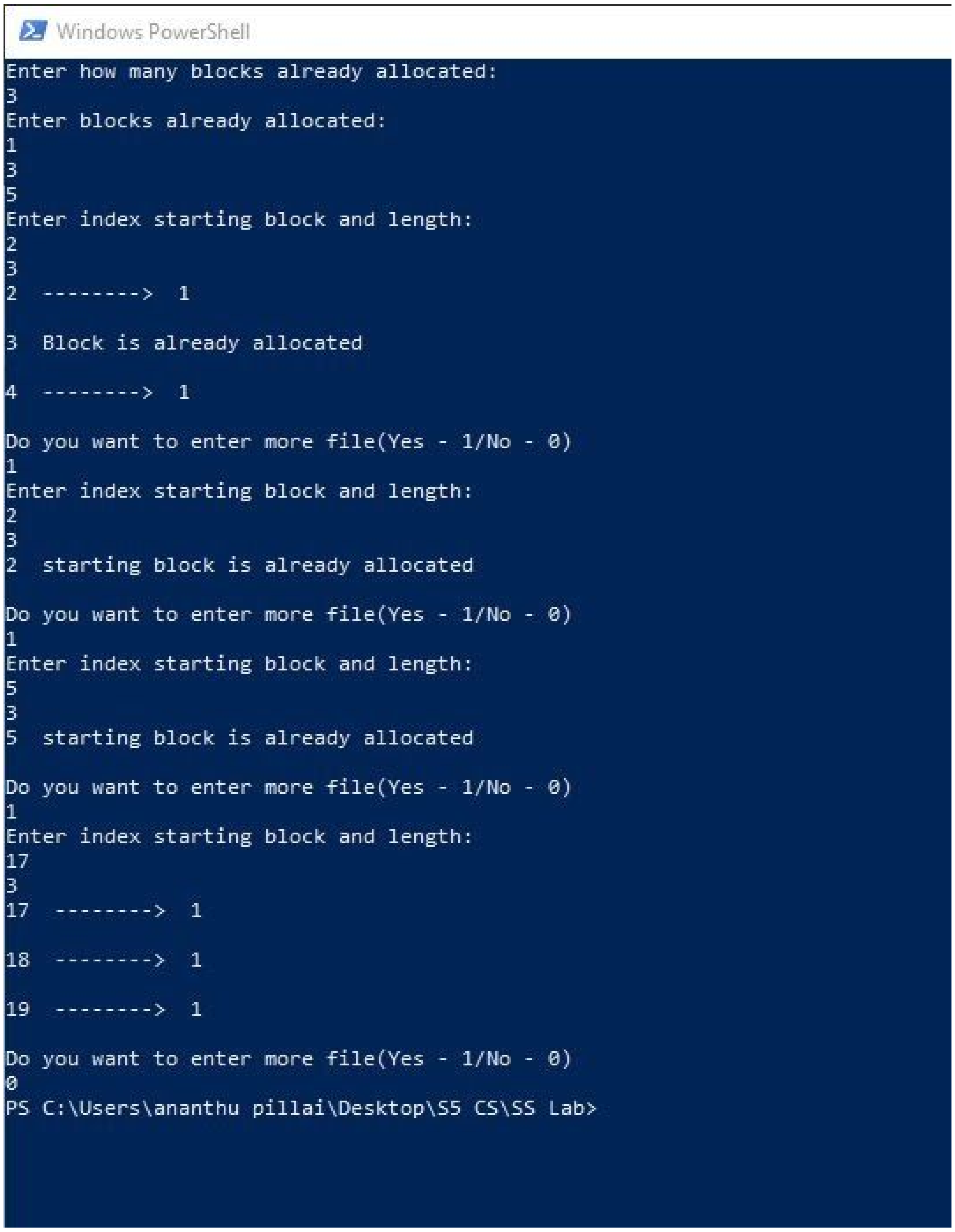
```
import os
def screen_clear():
   # for mac and linux(os.name is 'posix')
   if os.name == 'posix':
      _ = os.system('clear')
   else:
      # for windows platform
      _ = os.system('cls')
def main():
    f=[]
    c=1
    screen_clear()
    for i in range(50):
        f.append(0)
    print("Enter how many blocks already allocated: ");
    p=int(input())
    print("Enter blocks already allocated: ");
    for i in range(p):
        a=int(input())
        f[a]=1
    while c:
        print("Enter index starting block and length: ")
        st=int(input())
        len=int(input())
        k=len
        if(f[st]==0):
            for j in range(st,(st+k)):
                if(f[j]==0):
                    f[j]=1
                    print(j," --------> ",f[j],"\n")
                else:
                    print(j," Block is already allocated \n")
                    k=k+1
        else:
            print(st," starting block is already allocated \n")
        print("Do you want to enter more file(Yes - 1/No - 0)")
        c=int(input())
        if(c!=1):
            break

if __name__ == '__main__':
    main()
```



```python
import os
def screen_clear():
    # for mac and linux(here, os.name is 'posix')
    if os.name == 'posix':
        _ = os.system('clear')
    else:
        # for windows platfrom
        _ = os.system('cls')
def main():
    f=[]
    c=1
    screen_clear()
    for i in range(50):
        f.append(0)
    print("Enter how many blocks already allocated: ");
    p=int(input())
    print("Enter blocks already allocated: ");
    for i in range(p):
        a=int(input())
        f[a]=1
    while c:
        print("Enter index starting block and length: ")
        st=int(input())
        len=int(input())
        k=len
        if(f[st]==0):
            for j in range(st,(st+k)):
                if(f[j]==0):
                    f[j]=1
                    print(j," --------> ",f[j],"\n")
                else:
                    print(j," Block is already allocated \n")
                    k=k+1
        else:
            print(st," starting block is already allocated \n")
        print("Do you want to enter more file(Yes - 1/No - 0)")
        c=int(input())
        if(c!=1):
            break

if __name__ == '__main__':
    main()
```

Output:

```
Enter how many blocks already allocated:
3
Enter blocks already allocated:
1
3
5
Enter index starting block and length:
2
3
2  -------->  1

3  Block is already allocated

4  -------->  1

Do you want to enter more file(Yes - 1/No - 0)
1
Enter index starting block and length:
2
3
2  starting block is already allocated

Do you want to enter more file(Yes - 1/No - 0)
1
Enter index starting block and length:
5
3
5  starting block is already allocated

Do you want to enter more file(Yes - 1/No - 0)
1
Enter index starting block and length:
17
3
17  -------->  1

18  -------->  1

19  -------->  1

Do you want to enter more file(Yes - 1/No - 0)
0
```



```
Enter how many blocks already allocated:
3
Enter blocks already allocated:
1
3
5
Enter index starting block and length:
2
3
2  -------->  1

3  Block is already allocated

4  -------->  1

Do you want to enter more file(Yes - 1/No - 0)
1
Enter index starting block and length:
2
3
2  starting block is already allocated

Do you want to enter more file(Yes - 1/No - 0)
1
Enter index starting block and length:
5
3
5  starting block is already allocated

Do you want to enter more file(Yes - 1/No - 0)
1
Enter index starting block and length:
17
3
17  -------->  1

18  -------->  1

19  -------->  1

Do you want to enter more file(Yes - 1/No - 0)
0
PS C:\Users\ananthu pillai\Desktop\S5 CS\SS Lab>
```