PROGRAM: Sequential

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

struct file {
   char name[20];
   int sb, nob;
} f[30];

int blocks[100] = {0};
char allocated_files[100][20] = {{0}};

int main() {
   int i, j, n;

   printf("Enter no of files: ");
   scanf("%d", &n);

   for (i = 0; i < n; i++) {
      printf("\nEnter file name %d: ", i + 1);
      scanf("%s", f[i].name);
      printf("Enter starting block of file %d: ", i + 1);
      scanf("%d", &f[i].sb);
      printf("Enter no of blocks in file %d: ", i + 1);
      scanf("%d", &f[i].nob);

      for (j = 0; j < f[i].nob; j++) {
         if (blocks[f[i].sb + j] == 1) {
            printf("Block %d is already allocated to file %s. Please choose a different starting
block.\n", f[i].sb + j, allocated_files[f[i].sb + j]);
            i--;
            break;
         }
      }
      for (j = 0; j < f[i].nob; j++) {
         blocks[f[i].sb + j] = 1;
         strcpy(allocated_files[f[i].sb + j], f[i].name);
      }
   }
   printf("\nFILE NAME\tSTART BLOCK\tNO OF BLOCKS\tBLOCKS OCCUPIED\n");
   for(i=0;i<n;i++)
   {
      printf("\n%s\t\t%d\t\t%d\t\t", f[i].name, f[i].sb, f[i].nob);
```

DATE: 07/08/2024

EXPERIMENT NO. 2

FILE ALLOCATION STRATEGIES

AIM

Simulate the following file allocation strategies
a) Sequential b) Indexed c) Linked

ALGORITHM: Sequential

1. Start.
2. Initialize arrays for file properties and block allocation status.
3. Input the number of files.
4. For each file:
   a. Input the starting block and length.
   b. Check if the requested blocks are free:
      - If any block is already allocated, indicate failure for that file.
      - If all blocks are free, allocate them and mark them as used.
5. Display the allocation results for each file.
6. Implement a search loop:
   a. Prompt for a file number.
   b. If the file is allocated, display its details; otherwise, indicate it's not allocated.
   c. Ask if the user wants to search for another file.
7. End.

```
        for (j = 0; j < f[i].nob; j++)
            printf("%d, ", f[i].sb + j);
    }

    return 0;
}
```

OUTPUT

```
Enter no of files: 2

Enter file name 1: sample1
Enter starting block of file 1: 0
Enter no of blocks in file 1: 2

Enter file name 2: sample2
Enter starting block of file 2: 1
Enter no of blocks in file 2: 2
Block 1 is already allocated to file sample1. Please choose a different starting block.

Enter file name 2: sample2
Enter starting block of file 2: 11
Enter no of blocks in file 2: 5
```

FILE NAME	START BLOCK	NO OF BLOCKS	BLOCKS OCCUPIED
sample1	0	2	0, 1,
sample2	11	5	11, 12, 13, 14, 15,

PROGRAM: Indexed

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

struct file {
   char name[20];
   int nob;
   int blocks[30];
} f[30];

int allocated_blocks[30] = {0};
char allocated_files[30][20] = {{0}};

int main() {
   int i, j, n;
   char s[20];

   printf("Enter number of files: ");
   scanf("%d", &n);

   for (i = 0; i < n; i++) {
      printf("\nEnter file name %d: ", i + 1);
      scanf("%s", f[i].name);
      printf("Enter number of blocks in file %d: ", i + 1);
      scanf("%d", &f[i].nob);
      printf("Enter the blocks of the file: ");
      for (j = 0; j < f[i].nob; j++) {
         scanf("%d", &f[i].blocks[j]);
         if (allocated_blocks[f[i].blocks[j] - 1] == 1) {
            printf("Block %d is already allocated to file %s. Please choose a different block.\n",
f[i].blocks[j], allocated_files[f[i].blocks[j] - 1]);
            j--;
         } else {
            allocated_blocks[f[i].blocks[j] - 1] = 1;
            strcpy(allocated_files[f[i].blocks[j] - 1], f[i].name);
         }
      }
   }

   printf("\nFILE NAME\tNO OF BLOCKS\tBLOCKS OCCUPIED\n");
   for(i=0;i<f[i].nob;i++)
   {
      printf("%s\t\t%d\t\t", f[i].name, f[i].nob);
```

ALGORITHM: Indexed

1. START
2. Initialize arrays: allocated, fname, index, size, and block.
3. Set all elements of allocated to 0.
4. Input number of files n.
5. For each file i from 0 to n-1:
   a. Input file name fname[i].
   b. Input the number of blocks in the file f[i].nob.
   c. For each block j from 0 to f[i].nob - 1:
      - Input the block number f[i].blocks[j].
      - If allocated, prompt for another block.
   d. Print index block and allocated blocks.
6. Print the table of files with their index and size.
7. END

```
        for (j = 0; j < f[i].nob; j++) {
            printf("%d, ", f[i].blocks[j]);
        }
        printf("\n");
    }

    return 0;
}
```

OUTPUT

Enter number of files: 2

Enter file name 1: sample1

Enter number of blocks in file 1: 2

Enter the blocks of the file: 0 1

Enter file name 2: sample2

Enter number of blocks in file 2: 2

Enter the blocks of the file: 5 10

FILE NAME	NO OF BLOCKS	BLOCKS OCCUPIED
sample1	2	0, 1
sample2	2	5, 10

PROGRAM: Linked

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

struct file {
   char name[20];
   int nob;
   struct block {
      int bno;
      struct block *next;
   } *sb;
} f[30];

int allocated_blocks[30] = {0};
char allocated_files[30][20] = {{0}};

int main() {
   int i, j, n, retry;
   char s[20];
   struct block *temp, *prev;

   printf("Enter number of files: ");
   scanf("%d", &n);

   for (i = 0; i < n; i++) {
      retry = 0;
      printf("\nEnter file name %d: ", i + 1);
      scanf("%s", f[i].name);
      printf("Enter number of blocks in file %d: ", i + 1);
      scanf("%d", &f[i].nob);

      f[i].sb = (struct block *)malloc(sizeof(struct block));
      temp = f[i].sb;
      prev = NULL;
      printf("Enter the blocks of the file:");

      for (j = 0; j < f[i].nob; j++) {
         scanf("%d", &temp->bno);
         if (allocated_blocks[temp->bno - 1]) {
            printf("Block %d is already allocated to file %s. Please choose a different block.\n", temp-
>bno, allocated_files[temp->bno - 1]);
            temp = f[i].sb;
            while (temp != NULL) {
```

ALGORITHM: Linked

1. START
2. Initialize Variables:
   a. allocated_blocks[30] to track allocated blocks (all initialized to 0).
   b. allocated_files[30][20] to store file names.
   c. f[30] as an array of file structures to store file information.
3. Input Number of Files:
   a. Prompt the user for the number of files n.
4. For Each File (i = 0 to n-1):
   a. Input File Name: Read f[i].name.
   b. Input Number of Blocks: Read f[i].nob.
   c. Allocate memory for the starting block:
      - Set f[i].sb to a newly allocated block.
   d. Initialize a temporary pointer temp to f[i].sb and a pointer prev to track previous blocks.

      Input Block Numbers:
      - For each block j from 0 to f[i].nob - 1:
        - Loop until a valid block is entered:
          - Read temp->bno.
          - If allocated_blocks[temp->bno - 1] is 1, print an error message indicating the block is already allocated.
          - Free previously allocated blocks if the current block is invalid.
          - Set allocated_blocks[temp->bno - 1] to 1 and copy f[i].name to allocated_files[temp->bno - 1].
          - If not the last block, allocate memory for the next block and update temp.
   e. If a block is already allocated, decrement i to retry for the current file.
5. Output Allocation Information:
   a. Print headers: "FILE NAME", "NO OF BLOCKS", "BLOCKS OCCUPIED".
   b. For each file i, print f[i].name, f[i].nob, and the list of block numbers by iterating through the linked blocks starting from f[i].sb.
6. END

```
            prev = temp;
            temp = temp->next;
            free(prev);
          }
          retry = 1;
          break;
        }

        allocated_blocks[temp->bno - 1] = 1;
        strcpy(allocated_files[temp->bno - 1], f[i].name);

        if (j < f[i].nob - 1) {
          temp->next = (struct block *)malloc(sizeof(struct block));
          temp = temp->next;
        } else {
          temp->next = NULL;
        }
      }
      if (retry) {
        i--;
        continue;
      }
    }

    printf("\nFILE NAME\tNO OF BLOCKS\tBLOCKS OCCUPIED\n");
    for(i=0;i<n;i++)
    {
      printf("%s\t\t%d\t\t", f[i].name, f[i].nob);

      temp = f[i].sb;
      while (temp) {
        printf("%d", temp->bno);
        temp = temp->next;
        if (temp) {
          printf(" -> ");
        }
      }
      printf("\n");
    }

    return 0;
}
```

OUTPUT

Enter number of files: 2

Enter file name 1: sample1

Enter number of blocks in file 1: 2

Enter the blocks of the file:1 2

Enter file name 2: sample2

Enter number of blocks in file 2: 5

Enter the blocks of the file:5 10 15 20 25

FILE NAME	NO OF BLOCKS	BLOCKS OCCUPIED
sample1	2	1 -> 2
sample2	5	5 -> 10 -> 15 -> 20 -> 25

RESULT

The programs have been executed successfully and the expected output has been obtained and verified.