OPERATING SYSTEM-CSA0401
PROGRAM 21 - 30

KRITHIYA G
192421150

21. Worst Fit Memory Allocation in C

main.c

```c
#include <stdio.h>

void worstFit(int blockSize[], int m, int processSize[], int n)
    {
    int allocation[n];

    // Initially no block is assigned to any process
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    // Pick each process and find the worst fit block
    for (int i = 0; i < n; i++) {
        int wstIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (wstIdx == -1 || blockSize[j] >
                    blockSize[wstIdx])
                    wstIdx = j;
            }
```

Output

```
Process No. Process Size    Block No.
   1        212     5
   2        417     2
   3        112     5
   4        426     Not Allocated


=== Code Execution Successful ===
```

22. Best Fit Memory Allocation in C

main.c

```c
#include <stdio.h>

void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];

    // Initially no block is assigned to any process
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    // Pick each process and find the best fit block
    for (int i = 0; i < n; i++) {
        int bestIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (bestIdx == -1 || blockSize[j] <
                    blockSize[bestIdx])
                    bestIdx = j;
            }
        }
```

Output

```
Process No. Process Size    Block No.
   1        212     4
   2        417     2
   3        112     3
   4        426     5


=== Code Execution Successful ===
```

23. First Fit Memory Allocation

main.c

```c
#include <stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n)
    {
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock No.\n");
```

Output

```
Process No. Process Size    Block No.
   1        212     2
   2        417     5
   3        112     2
   4        426     Not Allocated


=== Code Execution Successful ===
```

24. Demonstrate UNIX System Calls for File Management

main.c

```c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main() {
    int fd;
    char buffer[100];

    fd = open("sample.txt", O_RDONLY);
    if (fd < 0) {
        perror("Failed to open file");
        return 1;
    }

    read(fd, buffer, sizeof(buffer));
    printf("File Contents:\n%s\n", buffer);

    close(fd);
    return 0;
```

Output

```
Failed to open file: No such file or directory


=== Code Exited With Errors ===
```

25. I/O System Calls (fcntl, seek, stat, opendir, readdir)

main.c

```c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/stat.h>

int main() {
    struct stat fileStat;
    int fd = open("sample.txt", O_RDWR);
    if (fd == -1) {
        perror("open");
        return 1;
    }

    fcntl(fd, F_SETFL, O_APPEND); // Set append mode
    lseek(fd, 0, SEEK_END);       // Seek to end
    write(fd, "\nAppended line.", 15); // Write to end

    fstat(fd, &fileStat);
```

Output

```
open: No such file or directory


=== Code Exited With Errors ===
```

26. File Management Operations

main.c

```c
#include <stdio.h>

int main() {
    FILE *fp;
    char str[100];

    fp = fopen("file.txt", "w");
    fprintf(fp, "Operating Systems File Handling\n");
    fclose(fp);

    fp = fopen("file.txt", "r");
    fgets(str, 100, fp);
    printf("Read from file: %s", str);
    fclose(fp);

    remove("file.txt"); // Deletes the file

    return 0;
}
```

Output

```
Segmentation fault


=== Code Exited With Errors ===
```

27. Simulate ls UNIX Command

main.c

```c
#include <dirent.h>

int main() {
    DIR *dir;
    struct dirent *entry;

    dir = opendir(".");
    if (dir == NULL) {
        perror("opendir");
        return 1;
    }

    printf("Files in current directory:\n");
    while ((entry = readdir(dir)) != NULL)
        printf("%s\n", entry->d_name);

    closedir(dir);
    return 0;
}
```

Output

```
Files in current directory:
.
..
.bash_logout
.bashrc
.profile


=== Code Execution Successful ===
```

28. Simulate grep UNIX Command

main.c

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

int main() {
    FILE *fp = fopen("sample.txt", "r");
    char word[] = "hello";
    char line[100];

    if (!fp) {
        perror("fopen");
        return 1;
    }

    while (fgets(line, sizeof(line), fp)) {
        if (strstr(line, word))
            printf("%s", line);
    }

    fclose(fp);
```

Output

```
fopen: No such file or directory


=== Code Exited With Errors ===
```

29. Classical Process Synchronization Problem (Producer-Consumer using Semaphore)

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define SIZE 5
int buffer[SIZE];
int in = 0, out = 0;

sem_t empty, full;
pthread_mutex_t mutex;

void* producer(void* arg) {
    int item = 1;
    for (int i = 0; i < 5; i++) {
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);

```

Output

```
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5


=== Code Execution Successful ===
```

30. Thread Concepts: Create, Join, Equal, Exit

main.c

```c
#include <stdio.h>
#include <pthread.h>

void* threadFunc(void* arg) {
    printf("Thread is running with ID: %lu\n", pthread_self());
    pthread_exit(NULL);
}

int main() {
    pthread_t tid1, tid2;

    pthread_create(&tid1, NULL, threadFunc, NULL);
    pthread_create(&tid2, NULL, threadFunc, NULL);

    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);

    if (pthread_equal(tid1, tid2))
        printf("Threads are equal\n");
```

Output

```
Thread is running with ID: 136832521828032
Thread is running with ID: 136832513435328
Threads are not equal
Main thread exiting.


=== Code Execution Successful ===
```