

## WEEK 6

**Program 1) Write a program to communicate parent and child process with each other in such a way that whenever child writes something, parent process can read it. Consider mode of communication is through.**

a. pipe

### Source Code:

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

int main()
{
    int p[2];

    int returnstatus;

    char writing[2][25] = {"hello", "world"};
    char readmsg[25];

    returnstatus = pipe(p);

    if (returnstatus == -1)

    {
        printf("Pipe not created\n");
        return 1;
    }

    printf("Writing started: %s\n", writing[0]);
    write(p[1], writing[0], strlen(writing[0]) + 1);
    read(p[0], readmsg, sizeof(readmsg));
    printf("Reading from pipe - msg1: %s\n", readmsg);

    return 0;
}
```

## OUTPUT

The screenshot shows a terminal window in the VS Code interface. The tab bar at the top includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', **TERMINAL**, 'PORTS', and 'POSTMAN CONSOLE'. The terminal content is as follows:

```
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab$ cd .git/.vscode/ Manual1/ WEEK1/ WEEK2/ WEEK3/ WEEK4/ WEEK5/ WEEK6/ WEEK7/ WEEK8/
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab$ cd WEEK6
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ gcc 1a.c -o 1a -lpthread
./1a
Writing started: hello
Reading from pipe - msg1: hello
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ gcc 1a.c -o 1a -lpthread
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./1a
Writing started: hello
Reading from pipe - msg1: hello
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$
```

At the bottom of the terminal window, there is a status bar with icons and text: 'Ln 21, Col 14' and 'Spaces: 4'.

## **b) message passing**

### **Source Code:**

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
#include <unistd.h>

struct msg_buffer
{
    long msg_type;
    char msg_text[100];
} message;

void main()
{
    key_t key;
    int msgid;
    key = ftok("progfile", 65);
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.msg_type = 1;
    if (fork() == 0)
    {
        strcpy(message.msg_text, "Message from child");
        msgsnd(msgid, &message, sizeof(message), 0);
    }
    else
    {
        msgrcv(msgid, &message, sizeof(message), 1, 0);
        printf("Parent read : %s\n", message.msg_text);
        msgctl(msgid, IPC_RMID, NULL);
    }
}
```

## OUTPUT

The screenshot shows a terminal window in VS Code with the following output:

```
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ gcc 1b.c -o 1b -lpthread
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./1b
Parent read : Message from child
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./1b
Parent read : Message from child
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), PORTS, and POSTMAN CONSOLE. A sidebar on the right shows options for powershell and wsl. The status bar at the bottom showsLn 38, Col 1 Spaces: 4 UTF-8 CRLF {} C Go Live Prettier.

### c) shared memory

#### Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    key_t key;
    int shmid;
    char *shared_memory;
    key = ftok(".", 65);
    if (key == -1)
    {
        perror("ftok");
        exit(1);
    }
    shmid = shmget(key, 1024, 0666 | IPC_CREAT);
    if (shmid == -1)
    {
        perror("shmget");
        exit(1);
    }
    if (fork() == 0)
    {
        shared_memory = (char *)shmat(shmid, NULL, 0);
```

```

if (shared_memory == (char *)(-1))
{
    perror("shmat");
    exit(1);
}

strcpy(shared_memory, "Message from child process");
printf("Child wrote: %s\n", shared_memory);

shmdt(shared_memory);
}

else
{
    sleep(1);

    shared_memory = (char *)shmat(shmid, NULL, 0);
    if (shared_memory == (char *)(-1))
    {
        perror("shmat");
        exit(1);
    }

    printf("Parent read: %s\n", shared_memory);

    shmdt(shared_memory);
    shmctl(shmid, IPC_RMID, NULL);
}
return 0;
}

```

## OUTPUT:

A screenshot of a terminal window in Visual Studio Code. The terminal tab is selected at the top. The output shows a child-parent communication example. The child process writes "Message from child process" and the parent process reads it. This interaction repeats twice.

```
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ gcc 1c.c -o 1c -lpthread imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./1c Child wrote: Message from child process Parent read: Message from child process imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./1c Child wrote: Message from child process Parent read: Message from child process imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$
```

The status bar at the bottom shows: Ln 62, Col 1 (1193 selected) Spaces:4 UTF-8 CRLF {} C Go Live Prettier

**Program2) Write a program to implement the concept of Producer-Consumer problem using semaphores.**

**Source Code:**

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define BUFFER_SIZE 5
int buffer[BUFFER_SIZE];
int in = 0, out = 0;
sem_t empty;
sem_t full;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
void *producer(void *args)
{
    int item;
    for (int i = 0; i < 10; i++)
    {
        item = i + 1;
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
        printf("Producer produced %d at index %d\n", item, in);
        in = (in + 1) % BUFFER_SIZE;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
        sleep(1);
    }
    return NULL;
}
```

```

void *consumer(void *args)
{
    int item;
    for (int i = 0; i < 10; i++)
    {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        item = buffer[out];
        printf("Consumer consumed %d from index %d\n", item, out);
        out = (out + 1) % BUFFER_SIZE;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
        sleep(2);
    }
    return NULL;
}

int main()
{
    pthread_t prod, cons;
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);
    pthread_create(&prod, NULL, producer, NULL);
    pthread_create(&cons, NULL, consumer, NULL);
    pthread_join(prod, NULL);
    pthread_join(cons, NULL);
    sem_destroy(&empty);
    sem_destroy(&full);
    pthread_mutex_destroy(&mutex);
    return 0;
}

```

## OUTPUT:

A screenshot of a terminal window from a code editor. The terminal tab is selected at the top. The command `gcc 2.c -o 2 -lpthread` is run, followed by the execution of the resulting binary `./2`. The output shows a sequence of producer and consumer operations on an array, with indices ranging from 0 to 9.

```
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ gcc 2.c -o 2 -lpthread
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./2
Producer produced 1 at index 0
Consumer consumed 1 from index 0
Producer produced 2 at index 1
Consumer consumed 2 from index 1
Producer produced 3 at index 2
Producer produced 4 at index 3
Consumer consumed 3 from index 2
Producer produced 5 at index 4
Producer produced 6 at index 0
Consumer consumed 4 from index 3
Producer produced 7 at index 1
Producer produced 8 at index 2
Consumer consumed 5 from index 4
Producer produced 9 at index 3
Producer produced 10 at index 4
Consumer consumed 6 from index 0
Consumer consumed 7 from index 1
Consumer consumed 8 from index 2
Consumer consumed 9 from index 3
```

At the bottom of the terminal window, there are status indicators: Line 51, Col 35, Spaces: 4, UTF-8, CRLF, {}, C, Go Live, Prettier, and a copy icon.

**Program3) Write a program to implement the concept of Dining-Philosopher problem.**

**Source Code:**

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#define N 5
sem_t chopstick[N];
void *philosopher(void *num)
{
    int id = *(int *)num;
    while (1)
    {
        printf("Philosopher %d is thinking\n", id);
        sleep(rand() % 3 + 1);
        int left = id;
        int right = (id + 1) % N;
        if (sem_trywait(&chopstick[left]) == 0)
        {
            if (sem_trywait(&chopstick[right]) == 0)
            {
                printf("Philosopher %d is eating using chopsticks %d and %d\n", id, left, right);
                sleep(rand() % 2 + 1); // eating time
                sem_post(&chopstick[left]);
                sem_post(&chopstick[right]);
                printf("Philosopher %d has released chopsticks %d and %d\n", id, left, right);
            }
        }
    }
}
```

```

    else
    {
        sem_post(&chopstick[left]);
        printf("Philosopher %d released chopstick %d since right chopstick %d is unavailable\n",
               id, left, right);
    }
}

else
{
    printf("Philosopher %d couldn't pick left chopstick %d, will try again\n", id, left);
}

usleep(200); // small delay to reduce CPU usage
}

pthread_exit(NULL);
}

void main()
{
    pthread_t tid[N];
    int ids[N];
    srand(time(NULL));
    for (int i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);
    for (int i = 0; i < N; i++)
    {
        ids[i] = i;
        pthread_create(&tid[i], NULL, philosopher, &ids[i]);
    }
    for (int i = 0; i < N; i++)
        pthread_join(tid[i], NULL);
}

```

## OUTPUT:

A screenshot of a terminal window from a code editor. The terminal tab is selected at the top. The command `gcc 3.c -o 3 -lpthread` was run, followed by `./3`. The output shows a sequence of messages from four philosophers (0, 1, 2, 3) indicating their state (thinking or eating) and actions (picking up or releasing chopsticks). The哲学家们正在思考或用餐，同时在尝试拿起筷子时会遇到冲突并放弃。

```
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ gcc 3.c -o 3 -lpthread
imanshu_aturi@LAPTOP-953U2DJJ:/mnt/c/Users/Himanshu/Desktop/OS Lab/WEEK6$ ./3
Philosopher 0 is thinking
Philosopher 3 is thinking
Philosopher 2 is thinking
Philosopher 4 is thinking
Philosopher 1 is thinking
Philosopher 2 is eating using chopsticks 2 and 3
Philosopher 1 released chopstick 1 since right chopstick 2 is unavailable
Philosopher 1 is thinking
Philosopher 3 couldn't pick left chopstick 3, will try again
Philosopher 2 has released chopsticks 2 and 3
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 0 is eating using chopsticks 0 and 1
Philosopher 4 released chopstick 4 since right chopstick 0 is unavailable
Philosopher 4 is thinking
Philosopher 1 couldn't pick left chopstick 1, will try again
Philosopher 2 is eating using chopsticks 2 and 3
Philosopher 1 is thinking
Philosopher 0 has released chopsticks 0 and 1
Philosopher 0 is thinking
Philosopher 4 is eating using chopsticks 4 and 0
Philosopher 3 couldn't pick left chopstick 3, will try again
```

Ln 82, Col 1 (2223 selected) Spaces:4 UTF-8 CRLF {} C Go Live Prettier

## WEEK 7

**Program1) FIFO – First In First Out :** page which came first (i.e. oldest page) need to be moved out.

### Source Code:

```
#include <stdio.h>

void enqueue(int q[], int *front, int *rear, int v, int n)
{
    if (*rear - *front + 1 == n)
    {
        printf("Queue is full\n");
        return;
    }
    else
    {
        if (*front == -1)
            *front = 0;
        q[++(*rear)] = v;
    }
}

int dequeue(int q[], int *front, int *rear)
{
    int v;
    if (*front == -1)
    {
        printf("Queue is empty\n");
        return -1;
    }
    else
    {
        v = q[*front];
        (*front)++;
    }
}
```

```

    if (*front > *rear)

    {
        *front = -1;
        *rear = -1;
    } }

return v;

}

int isPresent(int q[], int front, int rear, int page)
{
    if (front == -1)

        return 0;

    for (int i = front; i <= rear; i++)
    {
        if (q[i] == page)

            return 1;
    }

    return 0;
}

void display(int q[], int front, int rear)
{
    if (front == -1)

    {
        printf("Queue is empty\n");

        return;
    }

    for (int i = front; i <= rear; i++)

        printf("%d ", q[i]);

    printf("\n");
}

void main()
{
    int n, m;

```

```

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

printf("Enter number of page requests: ");
scanf("%d", &m);

int pages[m];

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

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

int q[20];

int front = -1, rear = -1;

int pageFaults = 0;

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

{

    int page = pages[i];

    printf("Request for page %d -> ", page);

    if (!isPresent(q, front, rear, page))

    { pageFaults++;

        if (front == -1 || rear - front + 1 < n)

            enqueue(q, &front, &rear, page, n);

        else{

            dequeue(q, &front, &rear);

            enqueue(q, &front, &rear, page, n);

        }

        printf("Page Fault Frames: ");

    } else

        printf("Page Hit Frames: ");

        display(q, front, rear);

    }

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

printf("Total Page Hits = %d\n", m - pageFaults);

}

```

## OUTPUT:

The screenshot shows a terminal window with the following content:

```
PS C:\Users\Himanshu\Desktop\OS Lab> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK7\" ; if ($?) { gcc 1.c -o 1 } ; if ($?) { .\1 }

● Enter number of frames: 3
Enter number of page requests: 12
2 3 2 1 5 2 4 5 3 2 5 2
Request for page 2 -> Page Fault Frames: 2 3
Request for page 3 -> Page Fault Frames: 2 3
Request for page 2 -> Page Hit Frames: 2 3
Request for page 1 -> Page Fault Frames: 2 3 1
Request for page 5 -> Page Fault Frames: 3 1 5
Request for page 2 -> Page Fault Frames: 1 5 2
Request for page 4 -> Page Fault Frames: 5 2 4
Request for page 5 -> Page Hit Frames: 5 2 4
Request for page 3 -> Page Fault Frames: 2 4 3
Request for page 2 -> Page Hit Frames: 2 4 3
Request for page 5 -> Page Fault Frames: 4 3 5
Request for page 2 -> Page Fault Frames: 3 5 2

Total Page Faults = 9
Total Page Hits = 3
○ PS C:\Users\Himanshu\Desktop\OS Lab\WEEK7>
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), PORTS, and POSTMAN CONSOLE. On the right side, there is a sidebar with icons for powershell, wsl, and Code WEEK7. The bottom of the window has a blue footer bar with icons for GitHub, Go Live, and a bell.

**Program2) LRU – Least Recently Used :** page which is has not been used for longest Ame need to be moved out.

**Source Code:**

```
#include <stdio.h>

int isPresent(int frames[], int n, int page) {
    for (int i = 0; i < n; i++) {
        if (frames[i] == page)
            return 1;
    }
    return 0;
}

int findLRU(int time[], int n) {
    int min = time[0], pos = 0;
    for (int i = 1; i < n; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

void display(int frames[], int n) {
    for (int i = 0; i < n; i++) {
        if (frames[i] != -1)
            printf("%d ", frames[i]);
        else
            printf("- ");
    }
    printf("\n");
}

void main()
```

```

{
    int n, m;

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

    printf("Enter number of page requests: ");
    scanf("%d", &m);

    int pages[m];

    printf("Enter the page reference string: ");

    for (int i = 0; i < m; i++)
        scanf("%d", &pages[i]);

    int frames[n], time[n];

    int counter = 0, pageFaults = 0;

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

    for (int i = 0; i < m; i++)
    {
        int page = pages[i];
        printf("Request for page %d -> ", page);

        if (isPresent(frames, n, page))
        {
            for (int j = 0; j < n; j++)
            {
                if (frames[j] == page)
                    time[j] = ++counter;
            }
            printf("Page Hit Frames: ");
        }
        else
            pageFaults++;
    }

    printf("Total Page Faults: %d", pageFaults);
}

```

```

    }

else

{

pageFaults++;



int emptyPos = -1;

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

{

if (frames[j] == -1)

{

emptyPos = j;

break;

}

}

if (emptyPos != -1)

{

frames[emptyPos] = page;

time[emptyPos] = ++counter;

}else {

int pos = findLRU(time, n);

frames[pos] = page;

time[pos] = ++counter;

}

printf("Page Fault Frames: ");

}

display(frames, n);

}

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

printf("Total Page Hits = %d\n", m - pageFaults);

}

```

## OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE + v ... | [x] x  
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK7> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK7\" ; if ($?) { gcc 2.c -o 2 } ; if (?) { ./2 }  
Enter number of frames: 3  
Enter number of page requests: 12  
Enter the page reference string: 2 3 2 1 5 2 4 5 3 2 5 2  
Request for page 2 -> Page Fault Frames: 2 - -  
Request for page 3 -> Page Fault Frames: 2 3 -  
● Request for page 2 -> Page Hit Frames: 2 3 -  
Request for page 1 -> Page Fault Frames: 2 3 1  
Request for page 5 -> Page Fault Frames: 2 5 1  
Request for page 2 -> Page Hit Frames: 2 5 1  
Request for page 4 -> Page Fault Frames: 2 5 4  
Request for page 5 -> Page Hit Frames: 2 5 4  
Request for page 3 -> Page Fault Frames: 3 5 4  
Request for page 2 -> Page Fault Frames: 3 5 2  
Request for page 5 -> Page Hit Frames: 3 5 2  
Request for page 2 -> Page Hit Frames: 3 5 2  
  
Total Page Faults = 7  
Total Page Hits = 5  
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK7>
```

## WEEK 8

**Program1) Best Fit – block which is closes to the size of request is allocated i.e. the smallest hole that is big enough to allocate to the requesting program.**

**Source Code:**

```
#include <stdio.h>

int main()
{
    int b, p;

    printf("Enter number of free blocks available : ");
    scanf("%d", &b);

    int block[b];

    for (int i = 0; i < b; i++)
        scanf("%d", &block[i]);

    printf("Enter number of processes : ");
    scanf("%d", &p);

    int process[p];

    for (int i = 0; i < p; i++)
        scanf("%d", &process[i]);

    int allocated[p];

    for (int i = 0; i < p; i++)
        allocated[i] = -1;

    for (int i = 0; i < p; i++)
    {
        int bestIdx = -1;

        for (int j = 0; j < b; j++)
        {
            if (block[j] >= process[i])
            {
                if (bestIdx == -1 || block[j] < block[bestIdx])
                    bestIdx = j;
            }
        }
    }
}
```

```

    }
}

if (bestIdx != -1)
{
    allocated[i] = bestIdx;
    block[bestIdx] -= process[i];
}

for (int i = 0; i < p; i++)
{
    if (allocated[i] != -1)
        printf("%d - %d\n", process[i], allocated[i] + 1);
    else
        printf("%d no free block allocated\n", process[i]);
}

return 0;
}

```

## OUTPUT:

The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE + v ... | [ ] X
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK7> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK8\" ; if ($?) { gcc 1.
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK7> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK8\" ; if ($?) { gcc 1.
c -o 1 } ; if ($?) { .\1 }
Enter number of free blocks available : 5
100 500 200 300 600
Enter number of processes : 4
212 417 112 426
● 212 - 4
417 - 2
112 - 3
426 - 5
○ PS C:\Users\Himanshu\Desktop\OS Lab\WEEK8>
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and POSTMAN CONSOLE. A sidebar on the right lists 'powershell', 'wsl', and 'Code WEEK8'. The bottom of the window has a blue bar with icons for GitHub, Go Live, and a bell.

**Program 2) First Fit – start searching the list from beginning, take the first block whose size is greater than or equal to the requesting program size and allocate it to program.**

**Source Code:**

```
#include <stdio.h>

int main()
{
    int b, p;

    printf("Enter number of free blocks available : ");
    scanf("%d", &b);

    int block[b];
    for (int i = 0; i < b; i++)
        scanf("%d", &block[i]);

    printf("Enter number of processes : ");
    scanf("%d", &p);

    int process[p];
    for (int i = 0; i < p; i++)
        scanf("%d", &process[i]);

    int allocated[p];
    for (int i = 0; i < p; i++)
        allocated[i] = -1;

    for (int i = 0; i < p; i++)
    {
        for (int j = 0; j < b; j++)
        {
            if (block[j] >= process[i])
            {
                allocated[i] = j;
                block[j] -= process[i];
                break;
            }
        }
    }
}
```

```
    }  
}  
  
for (int i = 0; i < p; i++)  
{  
    if (allocated[i] != -1)  
        printf("%d - %d\n", process[i], allocated[i] + 1);  
    else  
        printf("%d no free block allocated\n", process[i]);  
}  
  
return 0;  
}
```

## OUTPUT:

The screenshot shows a terminal window in Visual Studio Code with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE + v ... | [ ] X
● PS C:\Users\Himanshu\Desktop\OS Lab\WEEK8> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK8\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { ./2 }
Enter number of free blocks available : 5
100 500 200 300 600
Enter number of processes : 4
212 417 112 426
● 212 - 2
417 - 5
112 - 2
426 no free block allocated
○ PS C:\Users\Himanshu\Desktop\OS Lab\WEEK8>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and POSTMAN CONSOLE. The TERMINAL tab is active. The right sidebar shows icons for powershell, wsl, and Code WEEK8, with Code WEEK8 selected.

**Program3) Worst Fit – block which is largest among all is allocated for the program.**

**Source Code:**

```
#include <stdio.h>

int main()
{
    int b, p;

    printf("Enter number of free blocks available : ");
    scanf("%d", &b);

    int block[b];
    for (int i = 0; i < b; i++)
        scanf("%d", &block[i]);

    printf("Enter number of processes : ");
    scanf("%d", &p);

    int process[p];
    for (int i = 0; i < p; i++)
        scanf("%d", &process[i]);

    int allocated[p];
    for (int i = 0; i < p; i++)
        allocated[i] = -1;

    for (int i = 0; i < p; i++)
    {
        int worstIdx = -1;
        for (int j = 0; j < b; j++)
        {

```

```

    if (block[j] >= process[i])
    {
        if (worstIdx == -1 || block[j] > block[worstIdx])
            worstIdx = j;
    }
}

if (worstIdx != -1)
{
    allocated[i] = worstIdx;
    block[worstIdx] -= process[i];
}
}

for (int i = 0; i < p; i++)
{
    if (allocated[i] != -1)
        printf("%d - %d\n", process[i], allocated[i] + 1);
    else
        printf("%d no free block allocated\n", process[i]);
}

return 0;
}

```

## OUTPUT:

The screenshot shows a terminal window in Visual Studio Code. The terminal tab is selected at the top. The command `cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK8"; if (?) { gcc 3.c -o 3 } ; if (?) { ./3 }` is run, followed by the input `Enter number of free blocks available : 5` and the output `100 500 200 300 600`. Then, the input `Enter number of processes : 4` and the output `212 417 112 426` are shown. A blue dot indicates the next command: `212 - 5`. The output `417 - 2` and `112 - 5` follows. Finally, the message `426 no free block allocated` is displayed. The terminal ends with the prompt `PS C:\Users\Himanshu\Desktop\OS Lab\WEEK8>`.