Report on

Operating System Mini Group Project

at

MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

(Computer Science Engineering)

Teammates:	**Roll no:**
Rohin Sabharwal	2021A1R032
Hari Priya Dutta	2021A1R046
Vivek Singh Wazir	2021A1R036
Ujjwal Shaan	2021A1R038

Faculty Guide:

Assistant Professor Saurabh Sharma

Acknowledgement

We take this opportunity to express our sincere gratitude to all those who helped us in various capacities in undertaking this project and devising the report.

We are privileged to express our sense of gratitude to our faculty guide **Asst. Professor Saurabh Sharma** whose unparalleled knowledge, moral fiber and judgment along with his know-how , was an immense support in completing this project.

We are also grateful to **Dr. Ashok Kumar**, Dean Academics for the brainwave and encouragement given.

We are also entitled to **Professor Ankur Gupta**, the director of MIET, for their consistent support and motivation which leads us to better future.

We take this opportunity to thank our friends for their cooperation and compliance.

Table Of Contents

HEADING **PAGE NO:**

Problem Statement 5

Implementing First, Best and Worst Fit Contiguous memory allocation techniques by keeping a free/busy list of jobs organized by memory location

ABSTRACT

The technique to control and coordinate the computer memory, to assign blocks to different running programs for the optimization of an entire system's performance, is called memory management. It resides in hardware of the Operating System (OS), and in applications and programs .

Memory management is the main part of the Operating System which is basically used to control or handle the primary memory. Processes move between the disk and the main memory, during the time of execution. It keeps track of every memory location. It also checks the memory requirement for the processes and allocates memory as per that requirement. It also makes decisions about when the memory is allocated to the process. It updates the status of the memory whenever memory gets freed.

First Fit Memory Allocation :- This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, the first job claims the first available memory with space more than or equal to its size.

Best Fit Memory Allocation :- This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the

given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently.

Worst Fit Memory Allocation:- In this allocation technique, the process traverses the whole memory and always searches for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

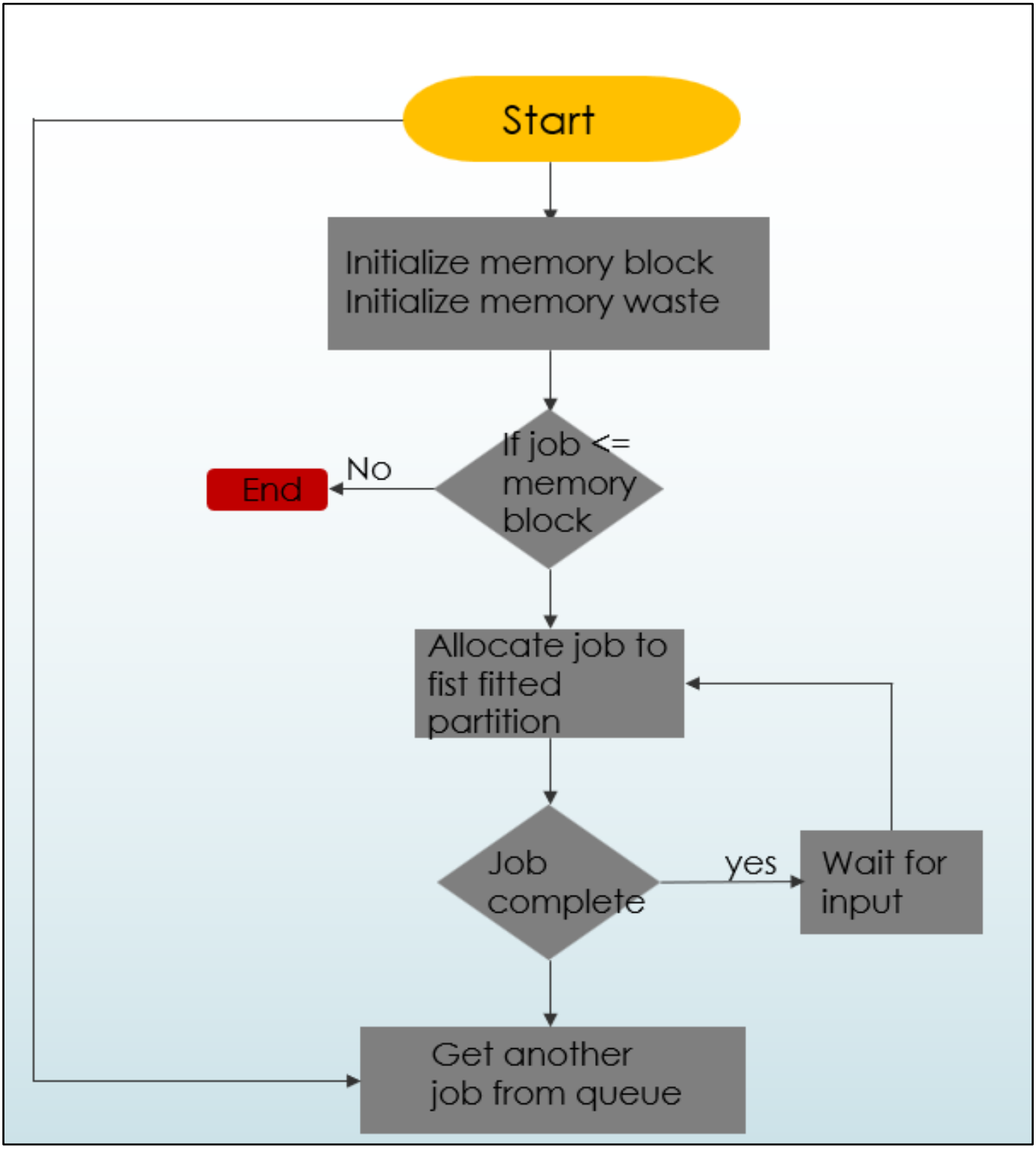
Objectives

- For both fixed and dynamic memory allocation schemes, the operating system must keep a list of each memory location noting which are free and which are busy.
- Then as new jobs come into the system, the free partitions must be allocated.
- These partitions may be allocated by **First-Fit Memory Allocation**, **Best-Fit Memory Allocation** and **Worst-Fit Memory Allocation**.
- Objective is to implement these allocation schemes on the Linux operating system .
- Users should be given a choice to select the particular allocation strategy.

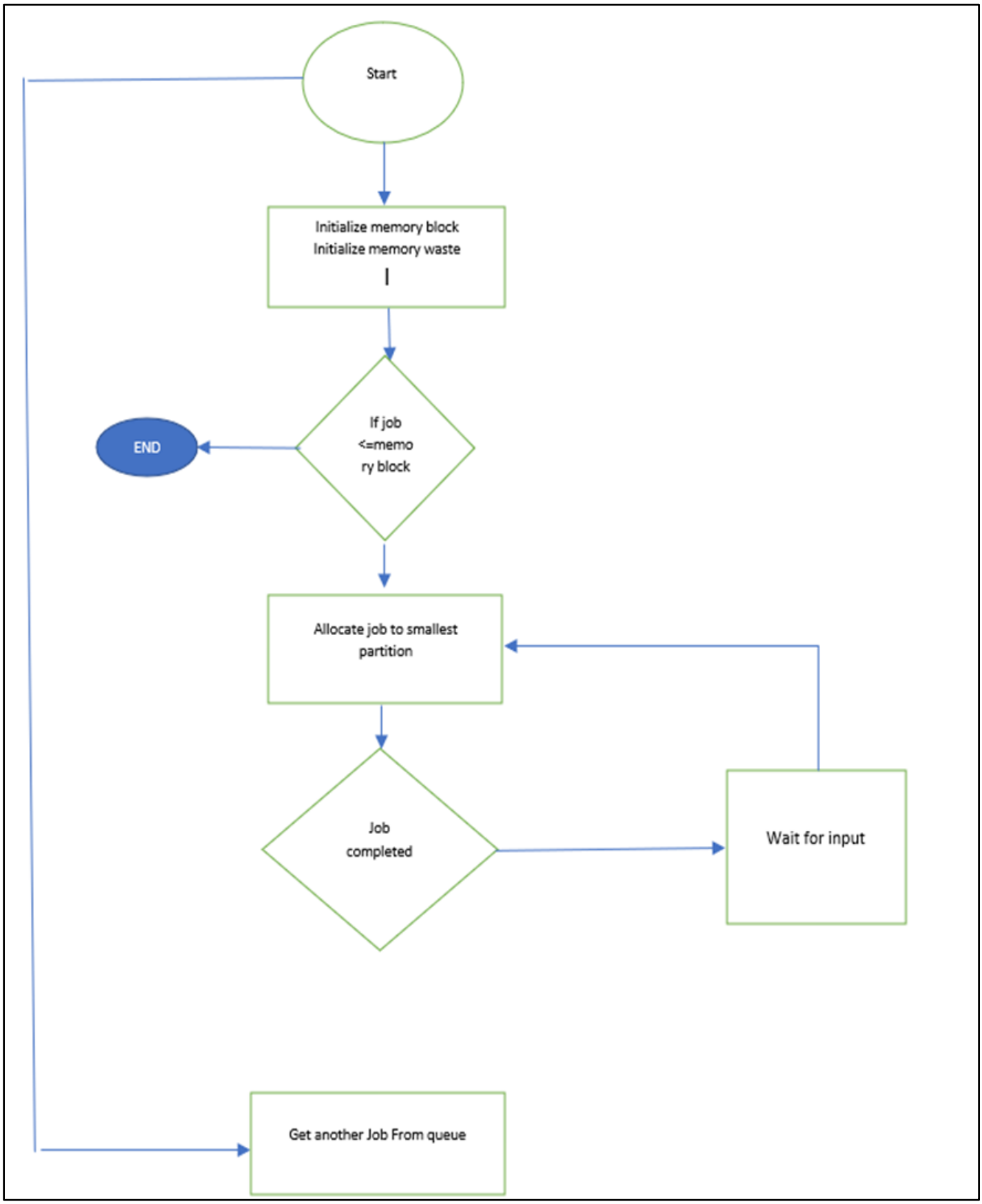
Methodology (flowchart)

Flowcharts for the problem statement are given:

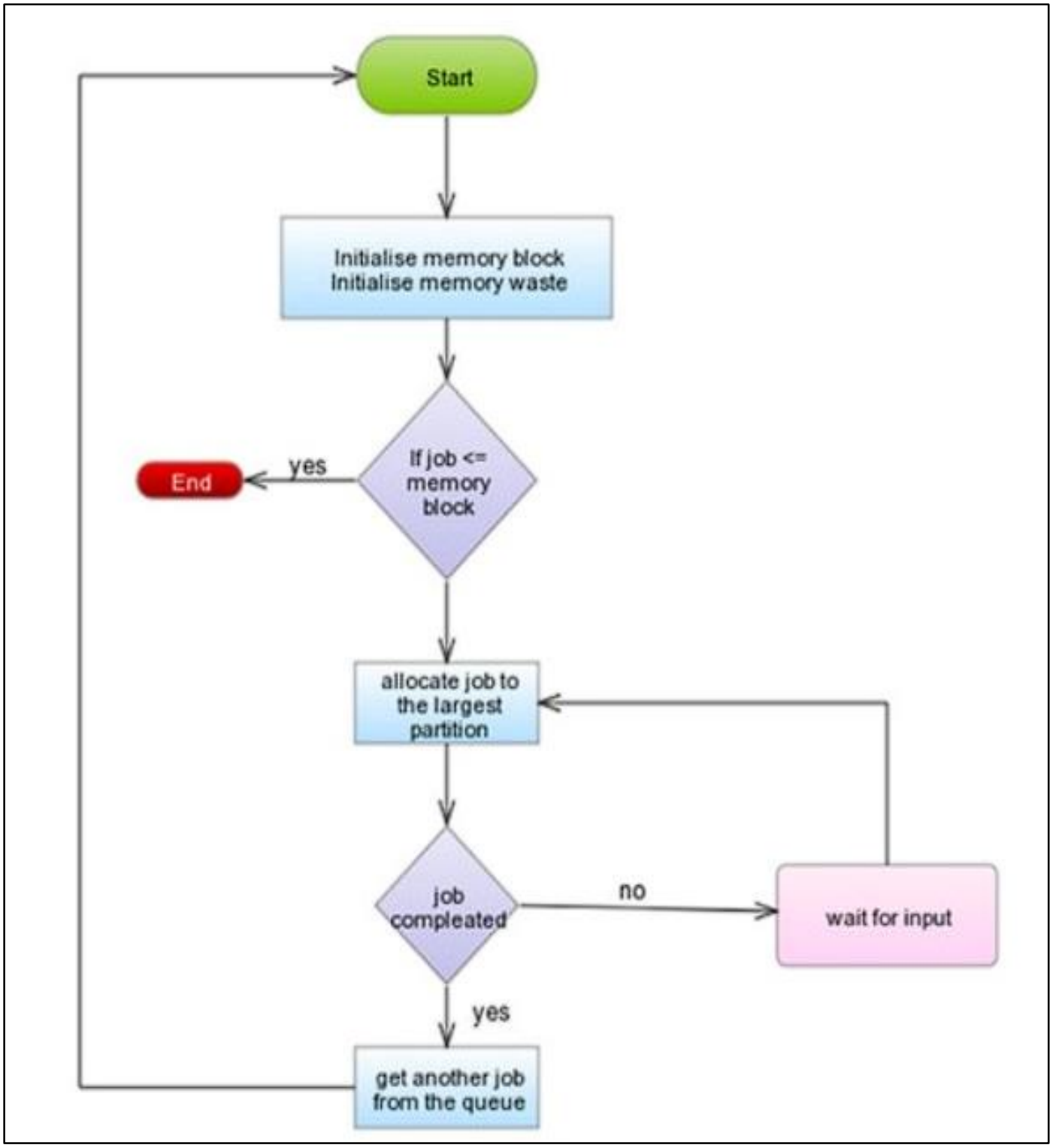
1)FIRST FIT:



2) BEST FIT:


Start
Initialize memory block
Initialize memory waste
If job <=memory block
END
Allocate job to smallest partition
Job completed
Wait for input
Get another Job From queue

3)WORST FIT:


Start
Initialise memory block
Initialise memory waste
If job <= memory block
yes
End
allocate job to the largest partition
job compleated
no
wait for input
yes
get another job from the queue

Algorithm

First Fit Memory Allocation Algorithm :

First-Fit Allocation In the first tit algorithm the allocator keeps a list of free blocks (known as the free list) and, on receiving a request for memory, scares along the list for the first block that is large enough to satisfy the request if the chosen block is significantly larger than that requested, then it is usually split, and the remainder added to the list as another free block The first fic algorithm performs reasonably well, as it ensures that allocations are quick When recycling free blocks there is a choice as to where to add the blocks to the free list effectively in what orders the free lot is kept

Algorithm for allocate (n)
Size(block)= n + size(header)
Scan free list for first block with nWords >=size(block)
If block not found
Failure (time for garbage collection!)
Else if free block nWords >= Free block nWords – size(block)
In-use block n words = size(block
Else
Unlink Block from free list
Return pointer to block
"Threshold must be at least size header) +1 to leave room for header and link
Threshold can be set higher to combat fragmentation
Allocation time is O(K) (K-number of Blocks in free list)

Best Fit Memory Allocation Algorithm:

1. Get no. of Processes and no. of blocks.

2. After that get the size of each block and process requests.

3. Then select the best memory block that can be allocated using the above definition.

4. Display the processes with the blocks that are allocated to a respective process.

5. Value of Fragmentation is optional to display to keep track of wasted memory.

6. Stop.

Worst Fit Memory Allocation Algorithm:

1. Input memory blocks and processes with sizes.

2. Initialize all memory blocks as free.

3. Start by picking each process and find the maximum block size that can be assigned to current process i.e., find max(bockSize[1], blockSize[2],.....blockSize[n]) > processSize[current],

4. If found then assign it to the current process.

5. If not then leave that process and keep checking the further processes.

Implementation

//C Program for Implementing First, Best and Worst Fir Contiguous memory allocation techniques by keeping free/busy list of jobs organized by memory location.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
 struct node {
   int start;
    int end;
    int size;
    struct node *next;
};
struct node *head = NULL;
struct node *tail = NULL;
_void insert(int start, int end, int size)
{ struct node*temp=structnode*)malloc(sizeof(struct node));
    temp->start = start;
    temp->end = end;
    temp->size = size;
    temp->next = NULL;
    if(head == NULL) {
        head = temp;
        tail = temp;  }
else {
        tail->next = temp;
        tail = temp; }}
void display() {
    struct node *temp = head;
    while(temp != NULL) {
        printf("%d\t%d\t%d\n", temp->start, temp->end, temp->size);
```

```
        temp = temp->next;   }}
void first_fit(int start, int end, int size)
{
    struct node *temp = head;
    while(temp != NULL)
    { if(temp->size >= size)
        {
            printf("\nJob Allocated at %d\n", temp->start);
            temp->start = temp->start + size;
            temp->size = temp->size - size;
            return;   }
        temp = temp->next; }
printf("\nJob Not Allocated\n"); }
void best_fit(int start, int end, int size)
{ struct node *temp = head;
    struct node *best = NULL;
    while(temp != NULL)
    {  if(temp->size >= size)
        { if(best == NULL)
            { best = temp;
            }
else if(best->size > temp->size)
            { best = temp;
            } }
        temp = temp->next; }
 if(best != NULL) {
        printf("\nJob Allocated at %d\n", best->start);
        best->start = best->start + size;
        best->size = best->size - size;  }
else   {
 printf("\nJob Not Allocated\n"); }
if(worst != NULL) void worst_fit(int start, int end, int size) {
```

```
  struct node *temp = head;
   struct node *worst = NULL;
    while(temp != NULL)
    {
        if(temp->size >= size)
        {
            if(worst == NULL)
            {
                worst = temp; }
}
            else if(worst->size < temp->size)
            {
                worst = temp;
            }
temp = temp->next;
    }

    {
        printf("\nJob Allocated at %d\n", worst->start);
        worst->start = worst->start + size;
        worst->size = worst->size - size;
    }
    Else
{
        printf("\nJob Not Allocated\n");
    }
}
int main()
{
    int start, end, size, choice;
    char ch;
    do  {
```

```
        printf("\nEnter the start, end and size of the memory block: ");
        scanf("%d%d%d", &start, &end, &size);
        insert(start, end, size);
printf("\nDo you want to enter more memory blocks? (y/n): ");
scanf(" %c", &ch);
    }while(ch == 'y' || ch == 'Y');
    printf("\nThe free/busy list of memory blocks is: \n");
    display();
 do  {
       printf("\nEnter the size of the job: ");
        scanf("%d", &size);
        printf("\n1. First Fit\n2. Best Fit\n3. Worst Fit\nEnter your choice: ");
        scanf("%d", &choice);
        switch(choice)  {
case 1: first_fit(start, end, size);
                    break;
case 2: best_fit(start, end, size);
                    break;
            case 3: worst_fit(start, end, size);
                    break;
            default: printf("\nInvalid Choice\n"); }
 printf("\nDo you want to enter more jobs? (y/n): ");
        scanf(" %c", &ch); }
while(ch == 'y' || ch == 'Y');
    printf("\nThe free/busy list of memory blocks is: \n");
   display();
    return 0;
}
```

(1)(2)

Test(Output)

The output of the C program is given below :

```
Enter the start, end and size of the memory block: 100 200 200

Do you want to enter more memory blocks? (y/n): y

Enter the start, end and size of the memory block: 200 300 100

Do you want to enter more memory blocks? (y/n): y

Enter the start, end and size of the memory block: 0 50 100
```

```
Do you want to enter more jobs? (y/n): y

Enter the size of the job: 50

1. First Fit
2. Best Fit
3. Worst Fit
Enter your choice: 2

Job Allocated at 200
```

```
Do you want to enter more jobs? (y/n): n

The free/busy list of memory blocks is:
111     200     189
250     300     50
0       50      100
rohin@rohin-VirtualBox:~$
```

```
Do you want to enter more memory blocks? (y/n): n

The free/busy list of memory blocks is:
100      200      200
200      300      100
0        50       100

Enter the size of the job: 1

1. First Fit
2. Best Fit
3. Worst Fit
Enter your choice: 1

Job Allocated at 100
```

```
Do you want to enter more jobs? (y/n): y

Enter the size of the job: 10

1. First Fit
2. Best Fit
3. Worst Fit
Enter your choice: 3

Job Allocated at 101
```

References

1.Geekforgeeks – https://www.geeksforgeeks.org/
2. Java point – https://www.javatpoint.com/
3. Quora -- https://www.quora.com/
4.Leetcode – https://www.quora.com/

Github

Below are the GitHub repository links of the teammates :

- https://github.com/haripriyadutta/2021a1r046___com-312
- https://github.com/rohin032/COM-312-PROJECT