Aim:

To study and implement Dynamic storage mapping strategies mainly

1.***First fit***

2.***Next Fit***

3.***Best Fit***

4.***Worst Fit***

Algorithm:

1.***First Fit***:

1.Initialize memory ,sequence array with the required number of blocks and sequences

2.After Initialization obtain the memory format /block size and store in the array

3.Iterate over the sequence array and the memory array from first and check if sequence[i]<=memory[j] if so set the flag as 1 and subtract memory[j]-=sequence[i] and break inner loop

4.If flag is 0 then it means there is no fit for the request in the memory at the present time and we indicate the same

5.While inside the iteration we print the mapping accordingly

2.***Next Fit***:

1.Initialize memory ,sequence array with the required number of blocks and sequences

2.After Initialization obtain the memory format /block size and store in the array

3.Iterate over the sequence array and the memory array from where the previous sequence was mapped and check if sequence[i]<=memory[j] if so set the flag as 1 and subtract memory[j]-=sequence[i] and break inner loop

4.If flag is 0 then it means there is no fit for the request in the memory at the present time and we indicate the same

5.While inside the iteration we print the mapping accordingly

3. ***Best Fit***:

1.Initialize memory, sequence array with the required number of blocks and sequences

2.After Initialization obtain the memory format /block size and store in the array

3.Sort the array (memory sequence) and store it in another variable say j

4.Now Iterate over the sequence array and the memory array and check if sequence[i]<=memory[j[i]] if so set the flag as 1 and subtract memory[j]-=sequence[i] and break inner loop

5.If flag is 0 then it means there is no fit for the request in the memory at the present time and we indicate the same

6.While inside the iteration we print the mapping accordingly

4.***Worst Fit:***

1.Initialize memory, sequence array with the required number of blocks and sequences

2.After Initialization obtain the memory format /block size and store in the array

3.Sort the array (memory sequence) and store it in another variable say j

4.Now Iterate over the sequence array and the memory array from reverse so as it is in descending order of space and check if sequence[i]<=memory[j[i]] if so set the flag as 1 and subtract memory[j]-=sequence[i] and break inner loop

5.If flag is 0 then it means there is no fit for the request in the memory at the present time and we indicate the same

6.While inside the iteration we print the mapping accordingly

Code:

dynamicStorageAlloc.c:

```
#include <stdio.h>
#include <stdlib.h>
void FirstFit(int * mem,int n_blocks,int* seq,int n_seq){
    int count_alloc=0,flag=0;
    for (int i=0;i<n_seq;i++){
        for(int j=0;j<n_blocks;j++){
            if(*(seq+i)<=*(mem+j)){
                printf("%d request sequence allocated to %d block\n",seq[i],j);
                *(mem+j)-=*(seq+i);
                flag=1;
                break;
            }
        }
        if(flag==0){
            printf("%d request sequence has to wait\n",*(seq+i));
```

```
                }
                flag=0;
        }
}
void NextFit(int * mem,int n_blocks,int* seq,int n_seq){
        int count_alloc=0,flag=0,c=0;
        int mem_ptr=0;
        for (int i=0;i<n_seq;i++){
                c=0;
                while(c<n_blocks){
                        if(*(seq+i)<=*(mem+mem_ptr)){
                                printf("%d request sequence allocated to %d
block\n",seq[i],mem_ptr);
                                *(mem+mem_ptr)-=*(seq+i);
                                flag=1;break;
                        }
                        mem_ptr++;
                        c++;
                        if(mem_ptr>n_blocks){mem_ptr=0;}
                }
                if(flag==0){
                        printf("%d request sequence has to wait\n",*(seq+i));
                }
                flag=0;
        }
}
int * sort(int *mem_sq,int size){
        int * arr=(int*)malloc(sizeof(int)*size);
        int* mem_seq=(int*)malloc(sizeof(int)*size);
        for(int i=0;i<size;i++){
                *(mem_seq+i)=*(mem_sq+i);
        }
        int ctr=0,tmp;
```

```
	for(int i=0;i<size;i++){
		for(int j=i+1;j<size;j++){
			if(*(mem_seq+i)>=*(mem_seq+j)){
				tmp=*(mem_seq+i);
				*(mem_seq+i)=*(mem_seq+j);
				*(mem_seq+j)=tmp;
			}
		}
	}
	return mem_seq;
}
void BestFit(int *mem1 ,int n_blocks,int *seq,int n_seq){
	int *mem=mem1;
	int *arr=sort(mem1,n_blocks);
	int c=0,flag=0;
	int * arr1=(int*)malloc(sizeof(int)*n_blocks);
	for(int i=0;i<n_blocks;i++){
		for(int j=0;j<n_blocks;j++){
			if(*(mem+i)==*(arr+j)){
				*(arr1+(c++))=j;break;
			}
		}
	}
	for(int i=0;i< n_seq;i++){
		for(int j=0;j<n_blocks;j++){
			if(*(seq+i)<=mem[*(arr1+j)]){
				printf("%d request sequence  allocated to %d
block\n",*(seq+i),*(arr1+(n_blocks-j)));
				*(mem+*(arr1+j))-=*(seq+i);
				flag=1;
				break;
			}
		}
```

```
		if(flag==0){
			printf("%d request sequence has to wait\n",*(seq+i));
		}
		flag=0;
	}
}
void WorstFit(int *mem1 ,int n_blocks,int *seq,int n_seq){
	int *mem=mem1;
	int *arr=sort(mem1,n_blocks);
	int c=0,flag=0;
	int * arr1=(int*)malloc(sizeof(int)*n_blocks);
	for(int i=0;i<n_blocks;i++){
		for(int j=0;j<n_blocks;j++){
			if(*(mem+i)==*(arr+j)){
				*(arr1+(c++))=j;break;
			}
		}
	}
	for(int i=0;i< n_seq;i++){
		for(int j=n_blocks-1;j>=0;j--){
			if(*(seq+i)<=mem[*(arr1+j)]){
				printf("%d request sequence  allocated to %d
block\n",*(seq+i),*(arr1+j));
				mem[arr1[j]]-=*(seq+i);
				flag=1;
				break;
			}
		}
		if(flag==0){
			printf("%d request sequence has to wait\n",*(seq+i));
		}
		flag=0;
	}
```

```
}
void memSet(int *mem1,int*mem2,int size){
	for(int i=0;i<size;i++){
		*(mem1+i)=*(mem2+i);
	}
}
int main(){
	int n_blocks,n_seq,assign_blocks,seq1;
	scanf("%d %d",&n_blocks,&n_seq);
	int * mem=(int *)malloc(n_blocks*sizeof(int));
	int*mem1=(int*)malloc(sizeof(int)*n_blocks);
	int*mem2=(int*)malloc(sizeof(int)*n_blocks);
	int*mem3=(int*)malloc(sizeof(int)*n_blocks);
	int* seq=(int*)malloc(n_seq*sizeof(int));
	printf("Enter the memory block sequence\n");
	for(int i=0;i<n_blocks;i++){
		scanf("%d",&assign_blocks);
		*(mem+i)=assign_blocks;
	}
	memSet(mem1,mem,n_blocks);
	memSet(mem2,mem,n_blocks);
	memSet(mem3,mem,n_blocks);
	printf("Enter the memory request sequence\n");
	for(int i=0;i<n_seq;i++){
		scanf("%d",&seq1);
		*(seq+i)=seq1;
	}
	printf("First Fit: \n");
	FirstFit(mem,n_blocks,seq,n_seq);
	printf("Next Fit: \n");
	NextFit(mem1,n_blocks,seq,n_seq);
	printf("Best Fit: \n");
```

```
    BestFit(mem2,n_blocks,seq,n_seq);
    printf("Worst Fit: \n");
    WorstFit(mem3,n_blocks,seq,n_seq);
}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise10# gcc dynamicStorageAlloc.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise10# ./a.out
5 4
Enter the memory block sequence
100 500 200 300 600
Enter the memory request sequence
212 417 112 426
First Fit:
212 request sequence allocated to 1 block
417 request sequence allocated to 4 block
112 request sequence allocated to 1 block
426 request sequence has to wait
Next Fit:
212 request sequence allocated to 1 block
417 request sequence allocated to 4 block
112 request sequence allocated to 4 block
426 request sequence has to wait
Best Fit:
212 request sequence  allocated to 4 block
417 request sequence  allocated to 2 block
112 request sequence  allocated to 1 block
426 request sequence  allocated to 3 block
Worst Fit:
212 request sequence  allocated to 4 block
417 request sequence  allocated to 1 block
112 request sequence  allocated to 4 block
426 request sequence has to wait
```

Result:

Thus, the Above four Memory allocation Strategies were tested and implemented.