```
Script started on 2019-03-10 16:49:26+0530
praveen@praveen cat Allocation.c
#include<stdio.h>
#include<stdlib.h>
struct Frames
{
    int size;
    int status;
    struct Frames * next;
};
typedef struct Frames frames;
//Insert
void insert(frames **head, frames **tail, int stat, int sz)
{
    frames* new_node = (frames*) malloc (sizeof(frames));
    new_node->status=stat;
    new_node->size=sz;
    new_node->next=NULL;
    if((*head)==NULL)
    {
        (*head)=(*tail)=new_node;
    }
    else
    {
        (*tail)->next=new_node;
        (*tail)=new_node;
    }
}
//Best-Fit
int Best_Fit(frames **head, frames **tail, int pid, int s)
{
    int newsize;
    frames *temp = *head;
    frames *p=NULL;
    while(temp!=NULL)
    {
        if((temp->status==-1)&&(temp->size>=s))
        {
            if(p==NULL)
                p=temp;
            else
            {
                if((p->size)>(temp->size)){
                    p=temp;
                }
            }
        }
        temp=temp->next;
    }
    newsize=p->size-s;
    if(newsize>0)
    {
        frames *newnode=(frames*)malloc(sizeof(frames));
        newnode->next=p->next;
        p->next=newnode;
        p->status=pid;
        newnode->size=newsize;
        newnode->status=-1;
    }
    else if(newsize==0)
    {
        p->size=s;
        p->status=pid;
    }
    if(p==NULL)
    {
        return 0;
    }
    else return 1;
}
//Worst-Fit
void worst_fit(frames **head,frames **tail,int pid,int size)
{
    frames *temp=*(head);
    frames *p=NULL;
    while(temp!=NULL){
        if((temp->status==-1)&&(temp->size>=size)){
            if(p==NULL){
                p=temp;
            }

            else if((p->size) < (temp->size))
                p=temp;
        }
        temp=temp->next;
    }
    if(p->size > size){
        p->status=pid;
        frames *temp1=(frames*)malloc(sizeof(frames));
        temp1->size=(p->size)-size;
        temp1->status=-1;
        temp1->next=p->next;
        p->size=size;
        p->next=temp1;
    }
    else if(p->size == size){
        p->status=pid;
    }
    else
        printf("Cannot insert!\n");

}
//First-Fit
int first_fit(frames** head, int pid, int size)
{
    frames* temp=*head;
    int k,flag=1;
    frames* new_node = (frames*) malloc (sizeof(frames));
    new_node->status=-1;
    while(temp!=NULL &&  size>temp->size)
    {
        temp=temp->next;
    }
    if(temp==NULL)
    {
        flag=0;
    }
    else if(size!=temp->size)
    {
        k=(temp)->size-size;
        new_node->size=k;
        temp->status=pid;
        temp->size=size;
        new_node->next=temp->next;
        temp->next=new_node;
        printf("%d",temp->size);
    }
    else if(size==temp->size)
    {
        temp->status=pid;
        free(new_node);
    }
    return flag;
}
void dealloc(frames** head,int pid)
{
    frames* temp=*head;
    int flag=0;
    while(temp!=NULL)
    {
        if(temp->status==pid)
        {
            temp->status=-1;
            flag=1;
        }
        else
        {
            temp=temp->next;
        }
    }
    if(flag==0)
        printf("No such process was found!\n");
}
//Display
void print_list(frames ** head)
{
    frames * node=(*head);
    while (node!=NULL)
    {
        printf("\n\n%d\t%d\n",node->status,node->size);
        node=node->next;
    }
}
//Coalesce
void coalesce(frames** head)
{
    frames* temp= *head;
    frames* temp1=NULL;
    while(temp!=NULL)
    {
        if(temp->status==-1)
        {
            temp1=temp->next;
            while(temp1!=NULL && temp1->status==-1)
            {
                temp->size = (temp->size) + (temp1->size);
                temp->next=temp1->next;
                temp1=temp->next;
            }
            temp=temp1;
        }
        else
        {
            temp=temp->next;
        }
    }
```

```
}
int main()
{
    frames *free_head = NULL;
    frames *free_tail = NULL;
    //Memory
    printf("Enter Number of Partitions: ");
    int n;
    int ch, choice;
    scanf("%d",&n);
    int a,b,c,d;
    for( int i=0;i<n;i++ )
    {
        printf("Enter start address of frame %d: ",i);
        scanf("%d",&a);
        printf("Enter end address of frame %d: ",i);
        scanf("%d",&b);
        c = -1;
        d = (b) - (a);
        insert(&free_head,&free_tail,c,d);
    }
    printf("FREE POOL MEMORY");
    print_list(&free_head);
    printf("\n\n");
    do
    {
        printf("\nMenu:");
        printf("\n1.Entry/Allocate");
        printf("\n2.Exit/Deallocate");
        printf("\n3.Display");
        printf("\n4.Coalesce");
        printf("\n5.Exit");
        int k;
        printf("\nEnter Choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
            {
                printf("Enter size of process: ");
                int sz;
                scanf("%d",&sz);
                printf("Enter PID: ");
                int p_pid;
                scanf("%d",&p_pid);
                do
                {
                    printf("\nMenu:");
                    printf("\n1.First Fit");
                    printf("\n2.Best Fit");
                    printf("\n3.Worst Fit");
                    printf("\n4.Exit");
                    printf("\nEnter Choice: ");
                    scanf("%d",&choice);
                    switch(choice)
                    {
                        case 1:
                        {
                            k=first_fit(&free_head,p_pid,sz);
                            if(!k) printf("\nProcess cannot be allocated!");
                            print_list(&free_head);
                            break;
                        }
                        case 2:
                        {
                            k=Best_Fit(&free_head,&free_tail,p_pid,sz);
                            if(!k) printf("\nProcess cannot be allocated!");
                            print_list(&free_head);
                            break;
                        }
                        case 3:
                        {
                            worst_fit(&free_head,&free_tail,p_pid,sz);
                            print_list(&free_head);
                            break;
                        }
                        case 4:
                        {
                            printf("\nExit!\n");
                            break;
                        }
                        default : printf("\nInvalid Choice!");
                    }
                }while(0);
                break;
            }
            case 2:
            {
                printf("\nEnter PID to deallocate: ");
                int p_pid;
                scanf("%d",&p_pid);
                dealloc(&free_head,p_pid);
                break;
            }
            case 3:
            {
                print_list(&free_head);
                break;
            }
            case 4:
            {
                coalesce(&free_head);
                print_list(&free_head);
                break;
            }
            case 5: {
                printf("\nExit!\n");
                break;
            }
            default : printf("\nInvalid Choice!");
        }
    }while(ch!=5);
    return 0;
}
```

```
praveen@praveen gcc Allocation.c
praveen@praveen ./a.out
Enter Number of Partitions: 10
Enter start address of frame 0: 0
Enter end address of frame 0: 120
Enter start address of frame 1: 121
Enter end address of frame 1: 270
Enter start address of frame 2: 271
Enter end address of frame 2: 300
Enter start address of frame 3: 301
Enter end address of frame 3: 400
Enter start address of frame 4: 401
Enter end address of frame 4: 600
Enter start address of frame 5: 601
Enter end address of frame 5: 620
Enter start address of frame 6: 621
Enter end address of frame 6: 720
Enter start address of frame 7: 721
Enter end address of frame 7: 900
Enter start address of frame 8: 901
Enter end address of frame 8: 1110
Enter start address of frame 9: 1111
Enter end address of frame 9: 1200
FREE POOL MEMORY

-1    120


-1    149


-1    29


-1    99


-1    199


-1    19


-1    99


-1    179


-1    209


-1    89



Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 1
Enter size of process: 98
Enter PID: 1

Menu:
1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Enter Choice: 2
```

```
-1    120


-1    149


-1    29


1     99


-1    1


-1    199


-1    19


-1    99


-1    179


-1    209


-1    89
Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 1
Enter size of process: 2
Enter PID: 2

Menu:
1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Enter Choice: 3


-1    120


-1    149


-1    29


1     99
```

```
-1    1


-1    199


-1    19


-1    99


-1    179


2     2


-1    207


-1    89
Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 1
Enter size of process: 200
Enter PID: 3

Menu:
1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Enter Choice: 1
200

-1    120


-1    149


-1    29


1     99


-1    1


-1    199


-1    19
```

```
-1    99


-1    179


2     2


3     200


-1    7


-1    89

Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 1
Enter size of process: 100
Enter PID: 4

Menu:
1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Enter Choice: 100

Invalid Choice!
Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 1
Enter size of process: 100
Enter PID: 4

Menu:
1.First Fit
2.Best Fit
3.Worst Fit
4.Exit
Enter Choice: 1
100

4     100


-1    20


-1    149
```

```
-1    29


1     99


-1    1


-1    199


-1    19


-1    99


-1    179


2     2


3     200


-1    7


-1    89
Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 2

Enter PID to deallocate: 4

Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 3


-1    100


-1    20


-1    149
```

```
-1    29


1     99


-1    1


-1    199


-1    19


-1    99


-1    179


2     2


3     200


-1    7


-1    89

Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
4.Coalesce
5.Exit
Enter Choice: 4


-1    298


1     99


-1    497


2     2


3     200


-1    96

Menu:
1.Entry/Allocate
2.Exit/Deallocate
3.Display
```

```
4.Coalesce
5.Exit
Enter Choice: 5

Exit!
praveen@praveen exit
exit

Script done on 2019-03-10 16:54:40+0530
```