Memory Management

```
#include<stdio.h>
#include<stdlib.h>


void display();
void best();
void worst();
void first();
void dealloc();
struct node
{
  int bid,free,allot,pno;
  struct node *link;
};
struct block
{
  int bid,diff;
}a[10];
int pid,space,b[10],p=0;
struct node *start,*trav;
main()
{
  int i,j=1000,ch;
  for(i=10;i>0;i--)
  {
    struct node *p;
    p=(struct node *)malloc(sizeof(struct node));
    p->bid=i;
    p->free=j;
    p->allot=0;
    p->link=start;
    start=p;
    j=j-100;
  }
  do
  {
    printf("\n1.Best Fit\n2.Worst Fit\n3.First
       Fit\n4.Deallocate\n5.Display\n6.Exit\n");
    printf("Enter Choice\n");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1: printf("\n Enter the process id:\n");
        scanf("%d",&pid);
        printf("\n Enter Space Required :\n");
        scanf("%d",&space);
```

```
        best();
        break;
      case 2: printf("\n Enter the process id:\n");
        scanf("%d",&pid);
        printf("\n Enter Space Required :\n");
        scanf("%d",&space);
        worst();
        break;
      case 3: printf("\n Enter the process id:\n");
        scanf("%d",&pid);
        printf("Enter Space Required:\n");
        scanf("%d",&space);
        first();
        break;
      case 4: printf("\n Enter the process id:\n");
        scanf("%d",&pid);
        dealloc();
        break;
      case 5: display();
        break;
      case 6: printf("Exiting\n");
        break;
      default: printf("Invalid Choice\n");

    }
  }while(ch!=6);
}
void display()
{
  int i,j,q;
  struct node *temp;

  temp=start;
  printf("\n");
  printf("\n\t ALLOTED-LIST \n");
  printf("\n BlockID  ProcessID   AllotedSpace\n");
  for(i=0;i<10;i++)
  {
    if((temp->allot)==0)
      temp=temp->link;
    else
    {
      printf("\n\t%d\t%d\t%d\n",temp->bid,temp->pno,temp->allot);
      temp=temp->link;
    }
  }
  temp=start;
  printf("\n\t FREE-LIST \n");
```

```
  printf("\n BlockID     FreeSpace\n");
  for(i=0;i<10;i++)
  {
    if(temp->free!=0)
      printf("\n\t%d\t%d\n",temp->bid,temp->free);
    temp=temp->link;
  }
  printf("\n");
}
void best()
{
  struct block t;
  int i,j,blid;
  trav=start;
  for(i=0;i<10;i++)
  {
    a[i].bid=trav->bid;
    if(space<=trav->free)
      a[i].diff=trav->free-space;
    else
      a[i].diff=-1;
    trav=trav->link;
  }
  for(i=0;i<10;i++)
    for(j=0;j<9;j++)
      if(a[j].diff>a[j+1].diff)
      {
        t=a[j];
        a[j]=a[j+1];
        a[j+1]=t;
      }
  i=9;
  if(a[i].diff==-1)
    printf("\nNo Sufficient Memory\n");
  else
  {
    for(i=0;i<10;i++)
      if(a[i].diff!=-1)
      {
        blid=a[i].bid;
        break;
      }
    printf("\n Process %d Fits in Block %d\n",pid,blid);
        trav=start;
    while(trav!=NULL)
    {
      if(trav->bid==blid)
      {
```

```
            trav->free=trav->free-space;
            trav->allot+=space;
            trav->pno=pid;
            break;
          }
          else
            trav=trav->link;
        }
        display();
    }
}
void worst()
{
    struct block t;
    int i,j,blid;
    trav=start;
    for(i=0;i<10;i++)
    {
      a[i].bid=trav->bid;
      if(space<trav->free)
        a[i].diff=trav->free-space;
      else
        a[i].diff=-1;
      trav=trav->link;
    }
    for(i=0;i<10;i++)
      for(j=0;j<9;j++)
        if(a[j].diff>a[j+1].diff)
        {
          t=a[j];
          a[j]=a[j+1];
          a[j+1]=t;
        }
    i=9;
    if(a[i].diff==-1)
      printf("\n No sufficient memory\n");
    else
    {
      blid=a[i].bid;
      printf("Process %d Fits in Block %d",pid,blid);
      trav=start;
      while(trav!=NULL)
      {
        if(trav->bid==blid)
        {
          trav->free=trav->free-space;
          trav->allot=trav->allot+space;
          trav->pno=pid;
```

```
            break;
          }
          else
            trav=trav->link;
        }
        display();
    }
}
void first()
{
    trav=start;
    while(trav!=NULL)
    {
        if(trav->free>=space)
        {
            trav->free=trav->free-space;
            trav->allot+=space;
            trav->pno=pid;
            printf("\n Process %d fits in block %d\n",pid,trav->bid);
            display();
            break;
        }
        else
            trav=trav->link;
    }
    if(trav==NULL)
    {
        printf("\nNo sufficient Memory\n");
    }
}
void dealloc()
{
    int i;
    struct node *temp;
    temp=start;
    for(i=0;i<10;i++)
    {
        if((temp->allot)==0)
            temp=temp->link;
        else
        {
            if(pid==temp->pno)
            {
                temp->free=temp->free+temp->allot;
                temp->allot=0;
            }
            temp=temp->link;
        }
```

```
    }
    display();
}
```

Output

```
42813@user:/mnt/42813/os$ gcc -g -o meman meman.c
42813@user:/mnt/42813/os$ ./meman

1.Best Fit
2.Worst Fit
3.First Fit
4.Deallocate
5.Display
6.Exit
Enter Choice
5


    ALLOTED-LIST

 BlockID   ProcessID    AllotedSpace

    FREE-LIST

 BlockID     FreeSpace

  1  100

  2  200

  3  300

  4  400

  5  500

  6  600

  7  700

  8  800

  9  900

  10   1000
```

```
1.Best Fit
2.Worst Fit
3.First Fit
4.Deallocate
5.Display
6.Exit
Enter Choice
1

 Enter the process id:
1

 Enter Space Required :
680

 Process 1 Fits in Block 7


   ALLOTED-LIST

 BlockID   ProcessID    AllotedSpace

  7 1 680

   FREE-LIST

 BlockID     FreeSpace

  1 100

  2 200

  3 300

  4 400

  5 500

  6 600

  7 20

  8 800

  9 900

  10   1000
```

```
1.Best Fit
2.Worst Fit
3.First Fit
4.Deallocate
5.Display
6.Exit
Enter Choice
2

 Enter the process id:
2

 Enter Space Required :
950
Process 2 Fits in Block 10

   ALLOTED-LIST

 BlockID   ProcessID    AllotedSpace

  7 1 680

  10   2 950

   FREE-LIST

 BlockID     FreeSpace

  1 100

  2 200

  3 300

  4 400

  5 500

  6 600

  7 20

  8 800

  9 900

  10   50
```

```
1.Best Fit
2.Worst Fit
3.First Fit
4.Deallocate
5.Display
6.Exit
Enter Choice
3

 Enter the process id:
3
Enter Space Required:
520

 Process 3 fits in block 6


   ALLOTED-LIST

 BlockID   ProcessID    AllotedSpace

  6 3 520

  7 1 680

  10   2 950

   FREE-LIST

 BlockID     FreeSpace

  1 100

  2 200

  3 300

  4 400

  5 500

  6 80

  7 20

  8 800
```

```
  9  900

  10   50


1.Best Fit
2.Worst Fit
3.First Fit
4.Deallocate
5.Display
6.Exit
Enter Choice
4

 Enter the process id:
2


    ALLOTED-LIST

 BlockID   ProcessID    AllotedSpace

  6  3  520

  7  1  680

    FREE-LIST

 BlockID     FreeSpace

  1  100

  2  200

  3  300

  4  400

  5  500

  6  80

  7  20

  8  800

  9  900
```

```
  10   1000


1.Best Fit
2.Worst Fit
3.First Fit
4.Deallocate
5.Display
6.Exit
Enter Choice
6
Exiting
42813@user:/mnt/42813/os$
```