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

typedef struct
{
      char pid[10];
      int size;
      int psize;
      int *allot;
      int n;
}process;


typedef struct
{
      int mem;
      int fsize;
      int *check;
      int n;
}phymem;


typedef struct
{
      process *p;
      int n;
}table;


void insert(phymem *m,table *t,process *p)
{
      int flag=0;
      int free;
      int i,j;
      for(i=0;i<m->n;i++)
      {
                  if(m->check[i]==0)
                              free++;
      }
      if(free>=p->size)
      {
                  for(i=0;i<p->n;i++)
                  {
                              for(j=0;j<m->n;j++)
                              {
                                    if(m->check[j]==0)
                                    {
                                                m->check[j]=1;
                                                p->allot[i]=j;
                                                break;
                                    }
                              }
```

```
                }
                t->p[t->n]=*p;
                t->n++;
        }
        else
                printf("Not enough free frames\n");

}


void page(table *t)
{
        int i,j;
        for(i=0;i<t->n;i++)
        {
                printf("\n\nPage Table for %s\n",t->p[i].pid);
                for(j=0;j<t->p[i].n;j++)
                {
                        printf("Page %d  :  Frame %d\n",j,t-
>p[i].allot[j]);
                }
        }
}


void freef(phymem *m)
{
        int i;
        printf("\nFree Frame:\n");
        for(i=0;i<m->n;i++)
        {
                if(m->check[i]==0)
                        printf("%d ",i);
        }
}


void dealloc(phymem *m,table *t,char *a)
{
        int i,j,index=-1;
        for(i=0;i<t->n;i++)
        {
                if(strcmp(a,t->p[i].pid)==0)
                {
                        index=i;
                        for(j=0;j<t->p[i].n;j++)
                        {
                                m->check[t->p[i].allot[j]]=0;
                        }
                        break;
                }
        }

        if(index==-1)
```

```
	{
			printf("Pid not found\n");
	}
	else
	{
			for(i=index;i<t->n-index;i++)
			{
				t->p[i]=t->p[i+1];
			}
			t->n--;
	}
}


void main()
{
	phymem *m;
	m=(phymem *)malloc(sizeof(phymem));

	printf("Enter physical memory size(KB) : ");
	scanf("%d",&m->mem);

	printf("Enter frame size(KB) : ");
	scanf("%d",&m->fsize);

	m->n=m->mem/m->fsize;
	printf("Physical memory is divided into %d frames\n",m->n);
	m->check=(int *)malloc(sizeof(int)*m->n);

	int i;
	for(i=0;i<m->n;i++)
			m->check[i]=0;

	srand(time(0));
	int j;
	for(i=0;i<m->n/2;i++)
	{
			j=rand()%m->n;
			m->check[j]=1;
	}

	printf("After Initialization\nFree Frames : ");
	for(i=0;i<m->n;i++)
	{
			if(m->check[i]==0)
				printf("%d ",i);
	}

	table *t;
	t=(table *)malloc(sizeof(table));
	t->p=(process*)malloc(sizeof(process)*10);
	t->n=0;

	int ch=1;
```

```
      while(ch!=5)
      {
                  printf("\n\n1.Process request\n2.Deallocation\n3.Page
Table\n4.Free Frame\n5.Exit\n");
                  scanf("%d",&ch);

                  switch(ch)
                  {
                              case 1:
                              {
                                    process p;

                                    printf("Process id  : ");
                                    scanf("%s",p.pid);

                                    printf("Requirement : ");
                                    scanf("%d",&p.size);
                                    p.psize=m->fsize;
                                    p.n=p.size/p.psize;
                                    p.n+=p.size%p.psize;
                                    printf("%s is divided into %d
pages\n",p.pid,p.n);
                                    p.allot=(int *)malloc(sizeof(int)*p.n);
                                    insert(m,t,&p);
                                    freef(m);
                                    break;
                              }
                              case 2:
                              {
                                    char a[10];
                                    printf("Enter pid to Deallocate : ");
                                    scanf("%s",a);
                                    dealloc(m,t,a);
                                    page(t);
                                    break;
                              }
                              case 3:
                              {
                                    page(t);
                                    break;
                              }
                              case 4:
                              {
                                    freef(m);
                                    break;
                              }
                              case 5:
                              {
                                    printf("Thank You\n");
                                    break;
                              }
                              default:
                                    printf("Invalid input\n");
                  }
```

```
      }
}



/*
Output:
Enter physical memory size(KB) : 32
Enter frame size(KB) : 1
Physical memory is divided into 32 frames
After Initialization
Free Frames : 0 1 2 5 6 7 8 10 11 12 13 14 15 16 19 20 22 25 30

1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
1
Process id  : p1
Requirement : 2
p1 is divided into 2 pages

Free Frame:
6. 5 6 7 8 10 11 12 13 14 15 16 19 20 22 25 30

1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
1
Process id  : p2
Requirement : 4
p2 is divided into 4 pages

Free Frame:
8 10 11 12 13 14 15 16 19 20 22 25 30

1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
1
Process id  : p3
Requirement : 3
p3 is divided into 3 pages

Free Frame:
12 13 14 15 16 19 20 22 25 30

1.  Process request
2.  Deallocation
```

```
3.  Page Table
4.  Free Frame
5.  Exit
3


Page Table for p1
Page 0  :  Frame 0
Page 1  :  Frame 1


Page Table for p2
Page 0  :  Frame 2
Page 1  :  Frame 5
Page 2  :  Frame 6
Page 3  :  Frame 7


Page Table for p3
Page 0  :  Frame 8
Page 1  :  Frame 10
Page 2  :  Frame 11


1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
4

Free Frame:
12 13 14 15 16 19 20 22 25 30

1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
2
Enter pid to Deallocate : p1


Page Table for p2
Page 0  :  Frame 2
Page 1  :  Frame 5
Page 2  :  Frame 6
Page 3  :  Frame 7


Page Table for p3
Page 0  :  Frame 8
Page 1  :  Frame 10
Page 2  :  Frame 11
```

```
1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
4

Free Frame:
0 1 12 13 14 15 16 19 20 22 25 30

1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
3


Page Table for p2
Page 0  :  Frame 2
Page 1  :  Frame 5
Page 2  :  Frame 6
Page 3  :  Frame 7


Page Table for p3
Page 0  :  Frame 8
Page 1  :  Frame 10
Page 2  :  Frame 11


1.  Process request
2.  Deallocation
3.  Page Table
4.  Free Frame
5.  Exit
5
Thank You
*/
```