**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<malloc.h>

void init();

void first();

void best();

void worst();

void dealloc();

void freelist();

void display();

void display1();

struct node

{

char id[20];

int mem;

struct node \*next;

}\*header,\*header1;

void main()

{

int c,n,i;

init();

printf("Enter The No.Of Free Nodes To Be Created\n");

scanf("%d",&n);

for(i=0;i<n;i++)

freelist();

while(1)

{

printf("\n\nMENU LIST\n");

printf("1.Create Free list\n2.First fit\n3.Best fit\n4.Worst fit\n

5.Deallocation\n6.Exit\n");

printf("Enter Your Choice\n");

scanf("%d",&c);

switch(c)

{

case 1:freelist();break;

case 2:first();break;

case 3:best();break;

case 4:worst();break;

case 5:dealloc();break;

case 6:exit(-1);

}

}

}

void init()

{

struct node \*temp,\*temp1;

temp=(struct node \*)malloc(sizeof(struct node));

header=temp;

temp1=(struct node \*)malloc(sizeof(struct node));

header1=temp1;

header->next=NULL;

header1->next=NULL;

}

void freelist()

{

struct node \*temp,\*ptr;

char ch[20];

int m;

char s[20];

while(1)

{

temp=(struct node \*)malloc(sizeof(struct node));

if(temp==NULL)

printf("\nInsufficient Memory\n\n");

else

{

printf("Enter the ID and Memory Size\n");

scanf("%s%d",s,&m);

strcpy(temp->id,s);

temp->mem=m;

temp->next=NULL;

if(header->next==NULL)

header->next=temp;

else

{

ptr=header->next;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next=temp;

}

}

printf("Do You Want To Continue? Y/N\n");

fflush(stdin);

scanf("%s",ch);

if(ch!="y")

break;

}

display();

display1();

printf("\n");

}

void display()

{

struct node \*ptr;

ptr=header->next;

printf("\nFree List : ");

if(ptr==NULL)

printf("Empty\n");

else

{

while(ptr!=NULL)

{

printf("%s %d ",ptr->id,ptr->mem);

ptr=ptr->next;

}

}

printf("\n");

}

void display1()

{

struct node \*ptr1;

ptr1=header1->next;

printf("Allocated List : ");

if(ptr1==NULL)

printf("Empty\n");

else

{

while(ptr1!=NULL)

{

printf("%s %d ",ptr1->id,ptr1->mem);

ptr1=ptr1->next;

}

}

printf("\n");

}

void first()

{

struct node \*temp,\*ptr,\*t,\*ptr1;

int m,flag=0;

printf("Enter The Required Memory\n");

scanf("%d",&m);

ptr=header->next;

while(ptr!=NULL)

{

if(ptr->mem>=m)

{

temp=(struct node \*)malloc(sizeof(struct node));

printf("Enter The ID\n");

scanf("%s",temp->id);

temp->mem=ptr->mem;

temp->next=NULL;

if(ptr==header->next)

header->next=ptr->next;

else

t->next=ptr->next;

if(header1->next==NULL)

header1->next=temp;

else

{

ptr1=header1->next;

while(ptr1->next!=NULL)

ptr1=ptr1->next;

ptr1->next=temp;

}

flag=1;

break;

}

t=ptr;

ptr=ptr->next;

}

if(flag==0)

printf("\nInsufficient Memory\n\n");

display();

display1();

}

void best()

{

struct node \*temp,\*ptr,\*t,\*ptr1;

int m,flag=0,n;

printf("Enter The Required Memory\n");

scanf("%d",&m);

for(ptr=header->next;ptr!=NULL;ptr=ptr->next)

{

for(ptr1=ptr->next;ptr1!=NULL;ptr1=ptr1->next)

{

if(ptr->mem > ptr1->mem)

{

n=ptr->mem;

ptr->mem=ptr1->mem;

ptr1->mem=n;

}

}

}

ptr=header->next;

while(ptr!=NULL)

{

if(ptr->mem>=m)

{

temp=(struct node \*)malloc(sizeof(struct node));

printf("Enter The ID\n");

scanf("%s",temp->id);

temp->mem=ptr->mem;

temp->next=NULL;

if(ptr==header->next)

header->next=ptr->next;

else

t->next=ptr->next;

if(header1->next==NULL)

header1->next=temp;

else

{

ptr1=header1->next;

while(ptr1->next!=NULL)

ptr1=ptr1->next;

ptr1->next=temp;

}

flag=1;

break;

}

t=ptr;

ptr=ptr->next;

}

if(flag==0)

printf("\nInsufficient Memory\n\n");

display();

display1();

}

void worst()

{

struct node \*temp,\*ptr,\*t,\*ptr1;

int m,flag=0,n;

printf("Enter The Required Memory\n");

scanf("%d",&m);

for(ptr=header->next;ptr!=NULL && (strcmp(ptr->id,"f")==0);ptr=ptr->next)

{

for(ptr1=ptr->next;ptr1!=NULL && (strcmp(ptr1->id,"f")==0);ptr1=ptr1->next)

{

if(ptr->mem < ptr1->mem)

{

n=ptr->mem;

ptr->mem=ptr1->mem;

ptr1->mem=n;

}

}

}

ptr=header->next;

while(ptr!=NULL)

{

if(ptr->mem>=m)

{

temp=(struct node \*)malloc(sizeof(struct node));

printf("Enter The ID\n");

scanf("%s",temp->id);

temp->mem=ptr->mem;

temp->next=NULL;

if(ptr==header->next)

header->next=ptr->next;

else

t->next=ptr->next;

if(header1->next==NULL)

header1->next=temp;

else

{

ptr1=header1->next;

while(ptr1->next!=NULL)

ptr1=ptr1->next;

ptr1->next=temp;

}

flag=1;

break;

}

t=ptr;

ptr=ptr->next;

}

if(flag==0)

printf("\nInsufficient Memory\n\n");

display();

display1();

}

void dealloc()

{

struct node \*ptr,\*temp,\*t,\*ptr1;

char s[20];

if(header->next==NULL && header1->next==NULL)

return;

else

{

ptr=header1->next;

printf("Enter the ID To Be Deallocated\n");

scanf("%s",s);

while(ptr!=NULL)

{

if(strcmp(ptr->id,s)==0)

{

if(ptr==header1->next)

header1->next=ptr->next;

else

t->next=ptr->next;

ptr1=header->next;

while(ptr1->next!=NULL)

ptr1=ptr1->next;

temp=(struct node \*)malloc(sizeof(struct node));

if(temp==NULL)

printf("\nInsufficient Memory\n\n");

else

{

temp->next=NULL;

ptr1->next=temp;

strcpy(temp->id,"f");

temp->mem=ptr->mem;

free(ptr);

}

break;

}

t=ptr;

ptr=ptr->next;

}

display();

display1();

}

}

**OUTPUT**

**student30@student30-Vostro-270s:~/Desktop/VANISHA46$ gcc mem.c**

**student30@student30-Vostro-270s:~/Desktop/VANISHA46$ ./a.out**

Enter The No.Of Free Nodes To Be Created

5

Enter the ID and Memory Size

f 100

Do You Want To Continue? Y/N

y

Free List : f 100

Allocated List : Empty

Enter the ID and Memory size

f 200

Do You Want To Continue? Y/N

y

Free List : f 100 f 200

Allocated List : Empty

Enter the ID and Memory Size

f 1000

Do You Want To Continue? Y/N

y

Free List : f 100 f 200 f 1000

Allocated List : Empty

Enter the ID and Memory Size

f 50

Do You Want To Continue? Y/N

y

Free List : f 100 f 200 f 1000 f 50

Allocated List : Empty

Enter the ID and Memory Size

f 900

Do You Want To Continue? Y/N

y

Free List : f 100 f 200 f 1000 f 50 f 900

Allocated List : Empty

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

2

Enter The Required Memory

50

Enter The ID

p1

Free List : f 200 f 1000 f 50 f 900

Allocated List : p1 100

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

3

Enter The Required Memory

70

Enter The ID

p2

Free List : f 50 f 900 f 1000

Allocated List : p1 100 p2 200

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

4

Enter The Required Memory

900

Enter The ID

p3

Free List : f 900 f 50

Allocated List : p1 100 p2 200 p3 1000

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

2

Enter The Required Memory

40

Enter The ID

p4

Free List : f 50

Allocated List : p1 100 p2 200 p3 1000 p4 900

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

5

Enter the ID To Be Deallocated

p2

Free List : f 50 f 200

Allocated List : p1 100 p3 1000 p4 900

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

1

Enter the ID and Memory size

f 500

Do You Want To Continue? Y/N

y

Free List : f 50 f 200 f 500

Allocated List : p1 100 p3 1000 p4 900

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

5

Enter the ID To Be Deallocated

p4

Free List : f 50 f 200 f 500 f 900

Allocated List : p1 100 p3 1000

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

3

Enter The Required Memory

100

Enter The ID

p5

Free List : f 50 f 500 f 900

Allocated List : p1 100 p3 1000 p5 200

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

4

Enter The Required Memory

400

Enter The ID

p6

Free List : f 500 f 50

Allocated List : p1 100 p3 1000 p5 200 p6 900

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

2

Enter The Required Memory

600

Insufficient Memory

Free List : f 500 f 50

Allocated List : p1 100 p3 1000 p5 200 p6 900

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

2

Enter The Required Memory

40

Enter The ID

p7

Free List : f 50

Allocated List : p1 100 p3 1000 p5 200 p6 900 p7 500

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

2

Enter The Required Memory

50

Enter The ID

p8

Free List : Empty

Allocated List : p1 100 p3 1000 p5 200 p6 900 p7 500 p8 50

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

5

Enter the ID To Be Deallocated

p6

Free List : f 900

Allocated List : p1 100 p3 1000 p5 200 p7 500 p8 50

MENU LIST

1.Create Free list

2.First fit

3.Best fit

4.Worst fit

5.Deallocation

6.Exit

Enter Your Choice

6