#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

char ssn[20], name[20], department[20], designation[20];

float sal;

long int phno;

struct node \*llink, \*rlink;

} NODE;

typedef struct headnode

{

int count;

struct node \*llink, \*rlink;

} HEAD;

void insfront(HEAD \*head);

void insrear(HEAD \*head);

void delfront(HEAD \*head);

void delrear(HEAD \*head);

void display(HEAD \*head);

NODE \*getNode();

void main()

{

int ch;

HEAD \*head = (HEAD \*) malloc(sizeof(HEAD));

head->count = 0;

head->llink = NULL;

head->rlink = NULL;

for(;;)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("1. Insert Front\n2. Insert Rear\n3. Delete Front\n4. Delete rear\n5.Display\n6. Exit\n");

scanf("%d", &ch);

switch(ch)

{

case 1:

insfront(head);

break;

case 2:

insrear(head);

break;

case 3:

if(head->rlink == NULL)

printf("List Empty");

else

delfront(head);

break;

case 4:

if(head->rlink == NULL)

printf("List Empty");

else

delrear(head);

break;

case 5:

if(head->rlink == NULL)

printf("List Empty");

else

display(head);

break;

case 6:

exit(0);

}

}

}

NODE \*getNode()

{

NODE \*temp = (NODE \*) malloc(sizeof(NODE));

if(temp == NULL)

{

printf("No Memory\n");

exit(0);

}

return temp;

}

void insfront(HEAD \*head)

{

NODE \*new = getNode();

NODE \*next = head->rlink;

printf("Enter Details such as\nSSN\nName\nDepartment\nDesignation\nSalary\nPhNo\n");

scanf("%s%s%s%s%f%ld", (new->ssn), (new->name), (new->department),

(new->designation), &(new->sal), &(new->phno));

if(next != NULL)

next->llink = new;

new->rlink = next;

head->rlink = new;

(head->count)++;

}

void insrear(HEAD \*head)

{

NODE \*new = getNode();

NODE \*temp = NULL;

printf("Enter Details such as SSN Name Department Designation Salary PhNo\n");

scanf("%s%s%s%s%f%ld", (new->ssn), (new->name), (new->department),

(new->designation), &(new->sal), &(new->phno));

(head->count)++;

new->rlink = NULL;

if(head->rlink == NULL)

{

head->rlink = new;

return;

}

temp = head->rlink;

while(temp->rlink != NULL)

temp = temp->rlink;

temp->rlink = new;

new->llink = temp;

}

void delfront(HEAD \*head)

{

NODE \*temp = head->rlink;

printf("Deleted Record is\n");

printf("%s\t%s\t%s\t%s\t%f\t%ld\n", (temp->ssn), (temp->name), (temp->department),

(temp->designation), (temp->sal), (temp->phno));

head->rlink = temp->rlink;

free(temp);

(head->count)--;

}

void delrear(HEAD \*head)

{

NODE \*previous = NULL, \*present = head->rlink;

if(present->rlink == NULL)

{

head->rlink = NULL;

}

else

{

while(present->rlink != NULL)

{

previous = present;

present = present->rlink;

}

previous->rlink = NULL;

}

printf("Deleted Record is\n");

printf("%s\t%s\t%s\t%s\t%f\t%ld\n", (present->ssn), (present->name), (present->department),

(present->designation), (present->sal), (present->phno));

(head->count)--;

free(present);

}

void display(HEAD \*head)

{

NODE \*temp = head->rlink;

printf("Total Number of records are %d\n", head->count);

printf("SSN\tName\tDepartment\tDesignation\tSalary\t\tPhNo\n");

while(temp != NULL)

{

printf("%s\t%s\t%s\t\t%s\t\t%f\t%ld\n", (temp->ssn), (temp->name),

(temp->department), (temp->designation), (temp->sal), (temp->phno));

temp = temp->rlink;

}

}