```
#include<stdio.h>             //including header files //stdio.h used for input or output operations
#include<string.h>                              //used for strcpy function
#include<stdlib.h>                              //used for system("clear") function
#include<termios.h>                              //used for getch function
#include<unistd.h>                              //used for getch function

struct dll                    //definition of the structure double linked list
{
char s[200];                               //character array
int index;
struct dll*prev,*next;              //pointers to next and previous nodes
};

int getch()                                  //getch function definition
{
struct termios oldt,
newt;
int ch;
tcgetattr( STDIN_FILENO, &oldt );
newt = oldt;
newt.c_lflag &= ~( ICANON | ECHO );
tcsetattr( STDIN_FILENO, TCSANOW, &newt );
ch = getchar();
tcsetattr( STDIN_FILENO, TCSANOW, &oldt );
return ch;
}


void editcommands(void);                        //function prototypes
void addline(struct dll *temp);
void inp(void);
void printlist(void);
void closer(void);
```

```c
void edit(void);
void addnode(char t[],struct dll *q);
void delnode(struct dll *p);
void clealist(void);
void editnode(struct dll *p);
void save(void);

struct dll *head;                         //header node declaration
char file[20];

FILE *fp=NULL;                             //file pointer declaration

main()
{
char c;

head=(struct dll*)malloc(sizeof(struct dll)); //header node memory allocation
 (head->next)=(head->prev)=NULL; //initialization
 (head->index)=0;

while(1)                //infinite while loop for editing multiple number of tiles
 {

system("clear");                         //clearing the screen

//Displaying editor options
printf("\nThis Editor provides the following options \n");
printf("R :opens a file and reads it into a buffer\n   If file doesnot exist creates a new file for editing\n");
printf("E :edit the currently open file\n");
printf("X :closes the current file and allows you to open another file\n");
printf("Q :quit discarding any unsaved changes\n");

c=getch();                             //taking user input
switch(c)                             //testing with switch
 {
case 'r' :
```

```
case 'R' :
inp();
break;
case 'e' :
case 'E' :
edit();
break;
case 'x' :
case 'X' :
closer();
break;
case 'q' :
case 'Q' :
system("clear");
exit(1);
break;
}
}
}                                //end of main



void addnode(char t[],struct dll *q)        //function to add a new node after a node q
{
struct dll*p=(struct dll*)malloc(sizeof(struct dll));
struct dll *temp=q->next;
strcpy(p->s,t);
p->prev=q;
p->next=q->next;

if((q->next)!=NULL)        //adding the node to the list by manipulating pointers accordingly
{
((q->next)->prev)=p;
while(temp!=NULL)
{
(temp->index)++;        //incrementing the index of the later nodes

temp=temp->next;
}
}
```

```
q->next=p;
p->index = q->index + 1;                    //setting the index of the new node
}



void delnode(struct dll *p)                    //function to delete a node
{
struct dll *temp=p->next;
((p->prev->next))=p->next;
if(p->next!=NULL)
{
((p->next)->prev)=p->prev;
while(temp!=NULL)
{
(temp->index)--;        //decrementing the index of the later nodes

temp=temp->next;
}
}
free(p);                          //freeing ht memory of the deleted node
}



void clearlist(void)                          //function to clear the list
{
while(head->next!=NULL)
delnode(head->next);                      //deleting all nodes except head
}



void editnode(struct dll *p)                    //function to edit a line
{
printf("\nThe original line is :\n%s",p->s);
printf("\nEnter the new line :\n");
gets(p->s);                              //taking the new line input
printf("\nLine edited\n");
}


void printlist(void)             //function to print all the lines stored in the buffer
{
struct dll *temp=head;
system("clear");
while(temp->next!=NULL)
{
temp=temp->next;
printf("%d %s\n",temp->index,temp->s);            //printing the lines on the screen
}
}



void closer(void)                //function to close the file orened for editing
{
if(fp==NULL)
return;
fclose(fp);
fp=NULL;
clearlist();                           //the list is also cleared at this point
}



void inp(void)
{
struct dll *buff=head;                          //temporaty variable
char c;
char buf[200];                                //array to store input line

```

```c
if(fp!=NULL)                              //checking for files opened earlier
{
printf("\nThere is another file open it will be closed\ndo you want to continue ?(Y/N):");
c=getch();
if(c=='n'||c=='N')
return;
else
closer();
}

fflush(stdin);
printf("\nEnter the file name you want to open :");
scanf("%s",file);
getchar();
fflush(stdin);
clearlist();

fp=fopen(file,"r");                              //opening the specified file
if(fp==NULL)                              //checking if the file previously exists
{
printf("\nThe file doesnot exist do you want to create one?(Y/N) :");
c=getchar();
getchar();
if(c=='N'||c=='n')
return;
else
{
fp=fopen(file,"w");                //creating new file
edit();
return;
}
}

if(feof(fp))
return;

while((fgets(buf,201,fp))!=NULL)                //taking input from file
```

```
{
addnode(buf,buff);
buff=buff->next;
}
edit();                              //calling the edit function
}



void void edit(void)                                 //the edit function
{
struct dll *temp=head->next;          //pionter used to mark the current position during traversing
char c,d;

system("clear");                                //clearing the screen

if(fp==NULL)                          //checking for files previously open
{
printf("\nNo file is open\n");
return;
}

printf("\nThe file contents will be displayed along with the line number\npress any key\n");
getch();
system("clear");
printlist();                                    //printing the entire buffered list
if(temp!=NULL)
printf("You are at line number %d",temp->index); //printing the line number of control
else
temp=head;

editcommands();                      //prints the list of commands available

while(1)                          //infinite loop for multiple command usage
```

```
{
c=getch();

switch(c)                          //switch -->condition checkig
{
case 'c' :
case 'C' :

editnode(temp);            //edit the current line pointed to by temp
break;

case 'p' :
case 'P' :                    //move to the previous line
if(temp==head)
{
printf("\nFile empty");    //message displayed if list is empty
break;
}
if(temp->prev!=head)
{
temp=temp->prev;
printf("\nYou are at line number %d",temp->index);
}
else                //message display if already at first line

printf("\nalready at first line");
break;

case 'n' :
case 'N' :                    //move to the next line
if(temp->next!=NULL)
{
temp=temp->next;
printf("\nYou are at line number %d",temp->index);
}
else if(temp==head)
printf("\nFile empty");        //message printed if list is empty
else
```

```
printf("\nalready at last line");//message printed if already at last line
break;

case 'a' :
case 'A' :             //adding a new line after node ponted by temp
addline(temp);                 //addline function takes input and creates a new node via addnode function
temp=temp->next;
printlist();
printf("\nYou are at line number %d",temp->index);
break;

case 'h' :
case 'H' :                    //HELP command displays the list of available commmands
system("clear");
editcommands();                //notice that there is no break as after help the entire list is printed
system("clear");

case 'v' :
case 'V' :                    //printing the entire list via printlist function
printlist();
printf("\nYou are at line number %d",temp->index);
break;

case 'D' :
case 'd' :                    //deleting a line pointed to by temp
if(temp==head)                //checking if list is empty
{
printf("\nFile empty\n");
break;
}
temp=temp->prev;
delnode(temp->next);            //deleting the node
printf("\nLine deleted\n");
printlist();                //printing the list
if(temp->index)
printf("\nYou are currently at line number %d",temp->index);
```

```
if(((temp->prev)==NULL)&&(temp->next)!=NULL)
temp=temp->next;
else if((temp==head)&&((temp->next)==NULL))
printf("\nFile empty");        //printing message if list is empty
break;

case 'X' :
case 'x' :                    //exit the editor to main menu

printf("\nDo you want to save the file before exiting?(y/n) :");

d=getch();                //warning for saving before exit
if(d=='y'||d=='Y')
save();
closer();
return;
break;

case 's' :
case 'S' :                    //saving and exitting
save();
closer();
return;
break;

}

}

}


void addline(struct dll *temp)                    //adding a new line via input from user
{
char buff[200];
printf("\nenter the new line :\n");
gets(buff);                          //taking the new line
addnode(buff,temp);                        //ceatng the new node
}
```

```


void save(void)                              //function to save the file
{
struct dll *temp=head->next;
fclose(fp);
fp=fopen(file,"w");                          //opening the file in write mode

while(temp!=NULL)
{
fprintf(fp,"%s",temp->s);                     //writing the linked list contents to file
temp=temp->next;
}

}


void editcommands(void)                          //function to print the list of editer commands
{
printf("\nEditor commands\n");
printf("The edit menu provides the following options \n");
printf("C :edit the current line\n");
printf("P :move one line up\n");
printf("N :move one line down\n");
printf("D :delete the current line\n");
printf("V :display the contents of the buffer\n");
printf("A :add a line after the line at which you are navigating\n");
printf("S :save changes and exit to main menu\n");
printf("X :exit discarding any changes\n");
printf("H :show the list of commands\n");
getch();
}
```