DATA STRUCTURES MINI PROJECT

NAME	ROLL NO	BATCH
SOHAM PRADHAN	16102B0003	1
MOHAMMED AAREM BARGE	16102B0006	1
ANIRUDDHA INGLE	16102B0016	1

INTRODUCTION

PROBLEM DEFINITION:

- To design a timetable using c programming language, in such a way that students could manage their homework and assignments by using the program.
- The time table should be designed in such a way that more preference should be given to subjects having submissions dates earlier.
- If the submission dates are same , more preference should be given to a particular subject that can be done in less time.
- looking at the date of submission and the total no of assignments or homework the time table should be able to tell the student the order in which he/she needs to do the work.
- The c program must use appropriate data structures for various options provided in the program.
- The program must display the time table ,insert tasks, delete tasks, sort, calculate reverse, split and merge the tasks.
- The program must display time table using basic graphics functions such as flood fill, setfillstyle,
  Rectangle , outtextxy.
- The program must compare the time recquired and the date of submission of two different assignments , and decide accordingly which task is to be completed first .

- While doing so the program must give priority to both the submission and time taken to complete and decide accordingly.

DECOMPOSING THE PROBLEM INTO MODULES:

1. DISPLAY THE TIMETABLE
    - Using graphics in C, we showcase the weekly timetable, once during the bulk insert of tasks.
    - With each and every insertion of task, the time table is displayed again for assistance in inserting the task.
2. INSERTING TASKS INTO THE PROGRAM:
    - The most basic problem that anyone would come across is accepting the tasks and storing them so that it can be used further in the program.
    - Just accepting is not an issue , the program is designed in such a way that each task that the user inputs utilizes minimum memory space and can be retained as long as required.
    - The task must not utilise permanent memory space that is the user should be able to delete it as and when required.
    - Taking all these into consideration we have designed a function insertend which inserts each task after the previous one (or the already present one).
    - This designed function insertend does not return any value so its return type is void , it is declared in the following manner:

void insertend(head*t , int ele , char name[ ] , int time);

- The uses of various parameters passed in function declaration are as follows:
  head*t – it is the pointer to structure method of accessing the start of the linked list.
  Int ele - it is a variable of data type integer , the date of submission of a particular task
  which is input by the user is stored in this variable.
  char name[ ] - It is a variable of datatype character , it is used to store the name of the
  assignment/ homework/task which is input by the user.
  Int time – time is a variable of datatype integer , it is used to store the value of time
  required to complete a particular task.

- The function insertend makes use of 2 pointers. One pointer is used to point to the new node that is created to store the values of new task that is to be inserted , the second pointer is used to traverse the linked list.
- The new node stores the date of submission , name of the task , time required to complete the task with the help of a pointer.
- If the linked list is empty , the entered task is made as task no 1 that is the node that contains the details of this task is made the first node of the linked list.
- If the linked list is not empty , new task is inserted at the end of this linked list .

3. DELETING THE FIRST TASK FROM THE PROGRAM:
   - Once a task is completed , there is a need to delete the task, known as delete scheduled task.

- Deleting the task is very essential because it frees up occupied memory thus making the program more efficient and fast to execute.
- There are different cases of deleting the tasks:
  1) Delete from beginning
  2) Delete from end
  3) Delete at position
  4) Delete element
- Delete from beginning: it is used the first node of the linked list. After deleting the First node , the second node is considered as first node.
- Delete from end : it is used to delete the last node . After deleting the last node , the Second last node is considered as the last node.
- Delete at position: it is used to delete the nodes that are present at a specific position Which is specified by the user .
- Delete element: this type of delete is used in linked list to delete the node which contains a particular element that the user has specified to be deleted.
- In our program we have used function deletebeg() (this is used to execute delete from beginning) and deleteele (this function is used to delete particular elements specified by the user).
- The deletebeg function prints the name of the task so its return type is character, it is declared in the following manner:

  void deletebeg(head *t);

- The uses of various parameters passed in function declaration are as follows:

head*t – it is the pointer to structure method of accessing the start of the linked list.

- This function deletebeg() utilizes one pointer which is used to traverse the linked list.
- If the linked list is empty the function prints (“ linked list is empty”) , and returns 0.
- If the linked list is not empty , the function returns the name of the first node .

4) DELETING A PARTICULAR TASK FROM THE PROGRAM:

- Once a task is completed , there is a need to delete the task.
- Deleting the task is very essential because it frees up occupied memory thus making the program more efficient and fast to execute.
- There are different cases of deleting the tasks:
  5) Delete from beginning
  6) Delete from end
  7) Delete at position
  8) Delete element
- Delete from beginning: it is used the first node of the linked list. After deleting the First node , the second node is considered as first node.
- Delete from end : it is used to delete the last node . After deleting the last node , the Second last node is considered as the last node.
- Delete at position: it is used to delete the nodes that are present at a specific position Which is specified by the user .
- Delete element: this type of delete is used in linked list to delete the node which contains a particular element that the user has specified to be deleted.

- In our program we have used function deletebeg() (this is used to execute delete from beginning) and deleteele (this function is used to delete particular elements specified by the user).
- The deleteele function returns the integer value so its return type is integer , it is declared in the following manner:

  void deleteele(head *t , int ele);

- The uses of various parameters passed in function declaration are as follows:
  head*t – it is the pointer to structure method of accessing the start of the linked list.
  Int ele - it is a variable of data type integer , the date of submission of a particular task
  which is input by the user is stored in this variable.
- This deleteele() function makes use of two pointers , one pointer is used to point to the node of the linked list that contains the element that has to be deleted and the other pointer is used to traverse the linked list .
- If the element that is to be deleted is present in the first node , one pointer points to this first node , start is made to point to the second node(this makes the second node as the starting node of the linked list) and then the first node is deleted.
- If the element to be deleted is not found at the first node of the linked list , a pointer is used to traverse the linked list so as to find the node which contains the element , then that node is deleted.
- If the element is not found , the function prints(“ element does not exist”).

5) SORTING THE LINKED LIST:

- In order to execute all other functions properly, it is essential to sort the nodes of the linked list.

- Sorting makes it very easy in terms of predicting and finding the location of any element that we wish to work on during the course of the program.
- We can sort the linked list in ascending or the descending order, as per our requirement.
- It helps in reducing the burden on computer's processor and memory that is complexity of any algorithm is reduced exponentially. Binary search is such algorithm which always demands that the array passed should be sorted.
- There are various algorithms that are used to for sorting the linked list:
  1) Bubble sort
  2) Quick sort
  3) Merge sort
  4) Insertion sort
  5) Selection sort
  6) Heap sort
  7) Bucket sort
  8) Radix sort
- In our program we have used the bubble sort technique to sort the elements of the linked list.
- The function sortll() in the program does the work of sorting the linked list in an ascending order.
- This function sortll() does not return anything , not even NULL , so it's return type is void, the syntax for this function is as follows:

```
void sortll(head *t);
```

- The uses of various parameters passed in function declaration are as follows:
  head*t – it is the pointer to structure method of accessing the start of the linked list.

- This bubble sort makes use of two pointers , one pointer is used to keep a check on the no of passes of the linked list and the other pointer is used to compare the data of nodes of the linked list.
- It makes use of a variable, in this case temp which is essentially required to execute the swapping if needed.
- Pointer 1 is initialised to first node and is incremented after one whole pass is completed.
- Pointer 2 is initialised to first node and it traverses and compares all nodes in each pass.
  If data of a node is greater than the data of next node then swapping takes place, if data is equal we consider data2 which is the duration taken to complete task. Larger duration is scheduled first, to get the larger task done first and out of the wa

6) REVERSING THE LINKED LIST:

- Reversing the linked list is essential when we want the program to display the priorities of tasks / assignments in the revere order.
- This function used in the program will display the least important / priority task first and the most important / priority task last.
- It gives the user the freedom to organise the tasks in either ascending or descending order.
- In our program we have used the function reverse () , to display the priorities of tasks in descending order.
- This function reverse () does not return anything so its return type is void.
- It is declared in the following manner in the program:

```
void reverse(head *t);
```

- The uses of various parameters used in the function declaration are as follows:
  head *t – it is the pointer to structure method to access the start of the linked list.
- This function uses 3 pointers, one pointer is initialised to start , one is made NULL, and the last one is not initialised at all.
- The pointer which is empty (the one which is not initialised at the start) should point to the second node of the linked list.
- The next of the first node of the linked list should be made NULL.
- The pointer which was initialised to NULL at the start should now point to the first node of the linked list.
- The pointer which was pointing to the first node of the linked list, now should point to the second node of the linked list where one pointer is already pointer.
- This whole process would continue while the pointer that was initialised to start is not equal to NULL.

7) SPLITTING THE LINKED LIST

- The above stated function allows the user to focus on a certain group of tasks, according to the date of submission.
- The function splits the linked list after a given date, this allows the user to focus on a more impending date of submission without being burdened by the entirety of the workload.
- This way the user gets to compartmentalise the workload and work efficiently towards finishing a task.

8) MERGING THE NODES OF THE LINKED LIST

- The functionality of this function is entirely based on the reversal of an error in the splitting of the linked list, it is an “undo” split function which is achieved using merge.

9) DISPLAY THE TASKS IN THE CORRECT ORDER:

- This function is used to display the tasks in order by which the user needs to complete them.

- This function is the most essential function of the program as the output of this function is displayed on the output screen.
- The output of the display function makes the program more interactive, and it gives the user the priorities of the tasks.
- In our program we have designed a function display() which is used to display the output.
- This function display does not return anything, so its return type is void.
- The function display() is declared in the following manner:

  void display(head *t)

- The uses of various parameters used in the function declaration are as follows:
  head *t – it is the pointer to structure method to access the start of the linked list.
- This function uses one pointer and a counter which is initialised to 1.
- If the linked list is empty , the function displays “linked list is empty.”
- The pointer points where start is pointing, that is to the first node of the linked list, and while it is not equal to NULL , display prints the name of the tasks in sorted order , ascending order.
- After each print the counter is initialised.

10) FUNCTION INPUT CLEAR BUFFER:

- Allows you to accept a string after an integer, it only gets character if it is not eof and not \n i.e Enter.
- It accepts and returns char, it creates a sort of a buffer, hence the ideal suited name for the function was, input clear buffer.
- The following while loop condition enables the clearing of the input buffer.

```
while (ch=getchar() !=EOF && ch!=”\n”)
{
        return(ch);
}
```

IDENTIFYING THE BEST SUITED DATA STRUCTURE

- There are various data structure that could have been used to implement the program.
- Some of the data structure are as follows:
  a. STACK
  b. QUEUES
  c. LINKED LIST
  d. TREES
  e. GRAPHS
- STACK:
  1) Stack is the most basic data structure.
  2) Stack can be imagined as a single ended open pipe where insertion and deletion can be done from one side only.
  3) Stack follows last in first out, “LIFO” approach.
  4) Stack was not a suitable data structure for our program because of various reasons.
  5) It is not easy to insert at the end of stack like in the case of queue’s and linked list.
  6) In order to implement insert end we need another stack , at first we need to pop the elements that already exist in stack and store them in other stack and then insert the element in stack 1 and reinsert all the elements from stack 2 to stack 1.
  7) Clearly this would require another function pop and an extra stack.
  8) Other functions as well cannot be implemented as easily by using stack.

- QUEUES:
  1) Like stack , queue is also a basic data structure.
  2) Queue can be imagined as a double ended open pipe , where insertion is done from rear end and deletion is done from the front end.
  3) Queue follows first in first out, “FIFO” approach.

4) While implementing insert end() is relatively easy in queue , other functions such as deleteele () , split() , merge () are not so easy to implement.
5) After a considerable number of insertions and deletions from a simple queue, it reaches it's saturation level.
6) Once a queue becomes saturated, every next insert results in queue overflow and evry next delete results in queue underflow.
7) Due to this queue is not a relevant data structure that could be used to implement the program.

- TREES:
    1) It is a collection of elements stored in non linear manner with a designated node called root from which the tree originates.
    2) Tree data structure is majorly used when we have to work with numbers.
    3) Functions like sort(), reverse(), split(), merge() are difficult to execute.
    4) Using trees in our program would increase the complexity.
    5) Hence tree is not a suitable data structure to be used.

- LINKED LIST:
    1) Linked list is the most suited data structure for this program.
    2) The main advantage of linked list is that, nodes can be added dynamically as and when required by us. It means that we do not have to predecide the size of linked list as in the case of arrays.
    3) It is very easy to delete and insert a node from a linked list.
    4) Traversing all nodes to display the linked list is easy as compared to traversing a node.
    5) Linked list can be easily sorted using any sorting technique.
    6) Any no of nodes can be inserted at the end of the linked list, insertend() function has been used extensively in our program.
    7) Starting node can be deleted easily using linked list. This makes the execution of deletebeg() easier.

8) Using linked lists, we can easily search for a node which contains a particular element and that node can be deleted without disturbing the order and sequence of other nodes.
9) The main advantage of using linked list is sorting.
   Bubble sort , bubble sort , merge sort , quick sort , heap sort can all be implemented using linked list.
10) Linked list can be printed in the reverse order with the help of a function defined in the program.

ALGORITHMS

- Function insertend()
  1) Start
  2) p ->data = ele
  3) p ->data2 = time
  4) Copy (p -> task, name)
  5) p ->next = NULL
  6) If (start = NULL)
     {
        Start = p
        Return
     }
  7) q = start
  8) while(q -> next != NULL)
     {
        q = q ->next
     }
  9) q - >next = p
  10) return

- function deletebeg: void
  1) Start
  2) If (start == NULL)
     {
        Print (" linked list is empty")
        Return
     }
  3) p = start
  4) copy (name , p -> task)

5) start = start -> next
6) print “p->task”

- function deleteele: void
  1) start
  2) if (start -> data == ele)
     {
       p = start
       start = start -> next
       print “p->task”
       return
     }
  3) q = start
  4) while (q ->next != NULL)
     {
       If (q ->next ->data = ele)
       {
         Break;
       }
       q = q - > next
     }
  5) if (q = = NULL)
     {
       Print “ element does not exist”
     }
  6) else
     {
       p = q ->next
       q ->next = p ->next
     }
  7) print “p->task”

- function sortll
    1) start
    2) for (I =start ; I -> next != NULL ; i++)
       {
          for (j = start ;j->next ! =NULL ; j++)
            {

                  if ((j ->data) > (j -> next -> data))
                   {
                       temp = j ->data
                       temp2=j->data2
                     Copy (str , j -> task)
                       j ->data = j ->next -> data
                       Copy (j ->task , j -> next-> task)
                       j -> next -> data = temp
                        j -> next -> data 2= temp2
                       copy (j -> next -> task , str)
                    }
             }
       }
    3) for (I =start ; I -> next != NULL ; i++)
       {
          for (j = start ;j->next ! =NULL ; j++)
            {

                  if (((j ->data)==(j -> next -> data))&&((j->data2)<(j->next->data2)))
                   {
                       temp = j ->data
                       temp2=j->data2
                     Copy (str , j -> task)
                       j ->data = j ->next -> data
                       Copy (j ->task , j -> next-> task)
                       j -> next -> data = temp
                        j -> next -> data 2= temp2

```
                    copy ( j -> next -> task , str)
                }
            }
        }
```

4) return

- function reverse
  1) start
  2) p = start
  3) q = NULL , r
  4) while (p! = NULL)
     {
     r = p -> next
     p -> next = q
     q =p
     p = r
     }
  5) start = q

- function display
  1) start
  2) i = 1
  3) if (start == NULL)
     {
     Print " linked list empty"
     Return
     }
  4) q = start
  5) print " to do task are:"
  6) while (q != NULL)
     {

Print “i++) q -> task to be submitted on q->data will take q->data2 hours”

q = q ->next

}

7) return

- function merge
  1) start
  2) q = start 1
  3) while (q ->next != NULL)
     {
     q = q ->next
     return
     }
  4) q -> next = start 2
  5) start 2 = NULL
  6) return

- function clr_ip_buf
  1) start
  2) int ch
  3) while ch=getchar() is not EOF and is not \n
  4) return ch;

CODE :

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include<dos.h>
```

```
typedef struct node
{
	char task[20];
	int data,data2;
	struct node *next, *prev;
}node;
typedef struct head
{
	node *start;
}head;

void display1();
void displaytable();
void displaytable2();
void insertend( head *t, int ele, char name[],int time);
void deletebeg( head *t);
void deleteele( head *t, char ele[]);
void sortll(head *t);
void reverse(head *t);
void split(head *t1, head *t2, int aft);
void merge(head *t1, head *t2);
void display(head *t);
int clr_ip_buf();


void display1()
{
	int gdriver = DETECT, gmode, errorcode;
	int left, top, right, bottom;
	int x,y,i,j=0;
	initgraph(&gdriver, &gmode, "../bgi");
	left=30;
	top=350;
	right=500;
	bottom=175;
```

```
        setfillstyle(SOLID_FILL,BLUE);
        rectangle(left,top,right,bottom);
        floodfill(10,10,WHITE);

        for(i=0;i<20;i++)
        {
                rectangle(left+j,top+j,right+j,bottom+j);
                j=j+3;
        }
        j=0;
        for(i=0;i<5;i++)
        {
        outtextxy( 125+j, 240+25+j,"PROJECT ON HOMEWORK MANAGEMENT:");
        outtextxy(125+j, 260+25+j, "\n SOHAM PRADHAN-16102B0003");
        outtextxy(125+j, 280+25+j,"\n AAREM BARGE-16102B0006");
        outtextxy(125+j,300+25+j,"\n ANIRUDDHA INGLE-16102B0016");
        j+=0.5;
        }
        j=0;
        printf("\n\n\n\n\n");
        system("pause");
        while(!kbhit())
        {

        }
        system("cls");
}

void main()
{
        head x,y;
        int gdriver = DETECT, gmode;
        int ch,ele,n,i,time;
        char name[20];
        x.start=NULL;
```

```
        y.start=NULL;
        initgraph(&gdriver, &gmode, "../bgi");
        display1();
        displaytable();
        printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
        printf("Enter No: of Initial Tasks:");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                printf("Enter Date of Submission MMDD for Task %d :\n",i);
                scanf("%d",&ele);
                clr_ip_buf();
                printf("Enter Name of Task:\n");
                gets(name);
                printf("Enter the time duration for the task in hours:\n");
                scanf("%d",&time);
                insertend(&x,ele,name,time);
        }
        while(1)
        {
                printf("\n1:InsertTask \n2:ScheduledTaskCompleted \n3:DeleteTask
\n4:ScheduleTasks\n");
                printf("5:Reverse \n6:Split \n7:Merge \n8:Display \n9:Exit\n");
                printf("Enter choice:\t");
                scanf("%d",&ch);
                printf("\n");
                if(ch==9)
                        break;
                switch(ch)
                {
                        case 1: displaytable2();
                                printf("Enter Date of Submission MMDD for Task %d
:\n",i);
                                scanf("%d",&ele);
                                clr_ip_buf();
```

```
            printf("Enter Name of Task:\n");
            gets(name);
            printf("Enter the Time Duration in Hours:\n");
            scanf("%d",&time);
            insertend(&x,ele,name,time);
            break;
    case 2:     if(x.start==NULL)
                printf("LL Empty\n");
            else
            {
                deletebeg(&x);
                display(&x);
            }
            break;
    case 3:     if(x.start==NULL)
                printf("LL Empty\n");
            else
            {
                clr_ip_buf();
                printf("Enter the task to be deleted:");
                gets(name);
                deleteele(&x,name);
                display(&x);
            }
            break;
    case 4: sortll(&x);
            display(&x);
            break;
    case 5:     reverse(&x);
            display(&x);
            break;
    case 6: printf("Enter the date MMDD you want to split after:");
            scanf("%d",&ele);
            split(&x,&y,ele);
            display(&x);
```

```
                        display(&y);
                        break;
                case 7: merge(&x,&y);
                        display(&x);
                        break;
                case 8:         display(&x);
                        break;
                default:printf("Invalid Operation \n");
            }
        }
  system("pause");
  getch();
  cleardevice();
  closegraph();
}

void displaytable()

  {
  int gdriver = DETECT, gmode, errorcode;
  int left, top, right, bottom;
  int x,y,i,j=0,k=50;
  initgraph(&gdriver, &gmode, "../bgi");
  left = 50;
  top = 50;
  right = 100;
  bottom = 100;
  setcolor(YELLOW);


  setfillstyle(SOLID_FILL,BLACK);
  rectangle(left,top,right+60,bottom);
  floodfill(51,51,YELLOW);

  for(i=0;i<3;i++)
```

```
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left,top+k,right+60,bottom+k);
	floodfill(51,51,YELLOW);
	k=k+50;
}

for(i=0;i<5;i++)
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left+110+j,top+50,right+50+60+j,bottom+50);
	floodfill(111+50,101,YELLOW);
	j=j+50;
}
j=0;

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168,125,"ECCF");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218,125,"DLDA");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268,125,"DSTR");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318,125,"DIM");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368,125,"AM 3");
floodfill(51,51,YELLOW);
```

```
for(i=0;i<5;i++)
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left+110+j,top+100,right+50+60+j,bottom+100);
	floodfill(111+50,101,YELLOW);
	j=j+50;
}
j=0;

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168,175,"AM 3");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218,175,"DIM");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268,175,"ECCF");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318,175,"DSTR");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368,175,"DLDA");
floodfill(51,51,YELLOW);

for(i=0;i<5;i++)
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left+110+j,top+150,right+50+60+j,bottom+150);
	floodfill(111+50,101,YELLOW);
```

```
        j=j+50;
}
j=0;

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168,225,"ECCF");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218,225,"DSTR");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268,225,"DIM");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318,225,"DLDA");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368,225,"AM 3");
floodfill(51,51,YELLOW);


setfillstyle(SOLID_FILL,BLACK);
outtextxy(65,75,"9.00-11.00");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(65,125,"11.15-1.15");
floodfill(51,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(65,175,"1.45-3.45");
```

```
  floodfill(51,51,YELLOW);

  setfillstyle(SOLID_FILL,BLACK);
  outtextxy(65,225,"3.45-5.45");
  floodfill(51,51,YELLOW);

  for(i=0;i<5;i++)
  {
        setfillstyle(SOLID_FILL,BLACK);
        rectangle(left+110+j,top,right+50+60+j,bottom);
        floodfill(51+50,51,YELLOW);
        j=j+50;
  }

  setfillstyle(SOLID_FILL,BLACK);
  outtextxy(168,75,"DSTR");
  floodfill(51,51,YELLOW);

  setfillstyle(SOLID_FILL,BLACK);
  outtextxy(218,75,"ECCF");
  floodfill(51,51,YELLOW);

  setfillstyle(SOLID_FILL,BLACK);
  outtextxy(268,75,"AM 3");
  floodfill(51,51,YELLOW);

  setfillstyle(SOLID_FILL,BLACK);
  outtextxy(318,75,"DIM");
  floodfill(51,51,YELLOW);

  setfillstyle(SOLID_FILL,BLACK);
  outtextxy(368,75,"DLDA");
  floodfill(51,51,YELLOW);
  }
void insertend( head *t, int ele, char name[],int time)
```

```
{
        node *p,*q;
        p=(node*)malloc(sizeof(node));
        p->data=ele;
        p->data2=time;
        strcpy(p->task,name);
        p->next=NULL;
        if(t->start==NULL)
        {
                t->start=p;
                return;
        }
        q=t->start;
        while(q->next!=NULL)
                q=q->next;
        q->next=p;
}
void deletebeg( head *t)
{
        node *p;
        if(t->start==NULL)
        {
                printf("LL Empty\n");
        }
        p=t->start;
        t->start=t->start->next;
        printf("Completed Task: %s \n",p->task);
}
void deleteele( head *t, char ele[])
{
        node *p,*q;
        if(strcmp(t->start->task,ele)==0)
        {
                p=t->start;
                t->start=t->start->next;
```

```
        }
        else
        {
                q=t->start;
                while(q->next!=NULL)
                {
                        if(strcmp(q->next->task,ele)==0)
                                break;
                        q=q->next;
                }
                if(q==NULL)
                        printf("Task Does Not Exist!!\n");
                else
                {
                        p=q->next;
                        q->next=p->next;
                }
        }
}
void sortll(head *t)
{
        node *i,*j;
        int t1,t2,str[20];
        for(i=t->start;i->next!=NULL;i=i->next)
        {
                for(j=t->start;j->next!=NULL;j=j->next)
                {
                        if((j->data)>(j->next->data))
                        {
                                t1=j->data;
                                t2=j->data2;
                                strcpy(str,j->task);
                                j->data=j->next->data;
                                j->data2=j->next->data2;
                                strcpy(j->task,j->next->task);
```

```
                        j->next->data=t1;
                        j->next->data2=t2;
                        strcpy(j->next->task,str);
                }

        }
    }
    for(i=t->start;i->next!=NULL;i=i->next)
    {
        for(j=t->start;j->next!=NULL;j=j->next)
        {
            if((j->next->data==j->data)&&(j->next->data2 > j->data2))
            {
                t1=j->data;
                t2=j->data2;
                strcpy(str,j->task);
                j->data=j->next->data;
                j->data2=j->next->data2;
                strcpy(j->task,j->next->task);
                j->next->data=t1;
                j->next->data2=t2;
                strcpy(j->next->task,str);
            }
         }
    }
}
void reverse(head *t)
{
    node *p=t->start,*q=NULL,*r;
    while(p!=NULL)
    {
        r=p->next;
        p->next=q;
        q=p;
        p=r;
```

```
        }
        t->start=q;
}
void split(head *t1, head *t2, int aft)
{
        node *q=t1->start;
        while(q->data!=aft)
                q=q->next;
        t2->start=q->next;
        q->next=NULL;

}
void merge(head *t1, head *t2)
{
        node *q=t1->start;
        while(q->next!=NULL)
                q=q->next;
        q->next=t2->start;
        t2->start=NULL;
}
void display(head *t)
{
        int i=1;
        node *q;
        if(t->start==NULL)
        {
                printf("LL Empty\n");
                return;
        }
        printf("\nStart of Task List\n");
        q=t->start;
        while(q!=NULL)
        {
```

```
                printf("%d) %s to be submitted on %d (MMDD) will take %d
hours\n",i++,q->task,q->data,q->data2);
                q=q->next;
        }
        printf("End of Task List\n\n");
}

int clr_ip_buf()
{
   int ch;
   while (((ch = getchar()) != EOF) && (ch != '\n'));
   return ch;
}

void displaytable2()

  {
  int gdriver = DETECT, gmode, errorcode;
  int left, top, right, bottom;
  int x,y,i,j=0,k=50;
  initgraph(&gdriver, &gmode, "../bgi");
  left = 250;
  top = 50;
  right = 300;
  bottom = 100;
  setcolor(YELLOW);


  setfillstyle(SOLID_FILL,BLACK);
  rectangle(left,top,right+60,bottom);
  floodfill(251,51,YELLOW);

  for(i=0;i<3;i++)
  {
        setfillstyle(SOLID_FILL,BLACK);
```

```
        rectangle(left,top+k,right+60,bottom+k);
        floodfill(251,51,YELLOW);
        k=k+50;
}

for(i=0;i<5;i++)
{
        setfillstyle(SOLID_FILL,BLACK);
        rectangle(left+110+j,top+50,right+50+60+j,bottom+50);
        floodfill(111+50+200,101,YELLOW);
        j=j+50;
}
j=0;

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168+200,125,"ECCF");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218+200,125,"DLDA");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268+200,125,"DSTR");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318+200,125,"DIM");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368+200,125,"AM 3");
floodfill(251,51,YELLOW);

for(i=0;i<5;i++)
```

```
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left+110+j,top+100,right+50+60+j,bottom+100);
	floodfill(111+50+200,101,YELLOW);
	j=j+50;
}
j=0;

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168+200,175,"AM 3");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218+200,175,"DIM");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268+200,175,"ECCF");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318+200,175,"DSTR");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368+200,175,"DLDA");
floodfill(251,51,YELLOW);

for(i=0;i<5;i++)
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left+110+j,top+150,right+50+60+j,bottom+150);
	floodfill(111+50+200,101,YELLOW);
	j=j+50;
}
```

```
j=0;

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168+200,225,"ECCF");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218+200,225,"DSTR");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268+200,225,"DIM");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318+200,225,"DLDA");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368+200,225,"AM 3");
floodfill(251,51,YELLOW);


setfillstyle(SOLID_FILL,BLACK);
outtextxy(65+200,75,"9.00-11.00");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(65+200,125,"11.15-1.15");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(65+200,175,"1.45-3.45");
floodfill(251,51,YELLOW);
```

```
setfillstyle(SOLID_FILL,BLACK);
outtextxy(65+200,225,"3.45-5.45");
floodfill(251,51,YELLOW);

for(i=0;i<5;i++)
{
	setfillstyle(SOLID_FILL,BLACK);
	rectangle(left+110+j,top,right+50+60+j,bottom);
	floodfill(51+50+200,51,YELLOW);
	j=j+50;
}

setfillstyle(SOLID_FILL,BLACK);
outtextxy(168+200,75,"DSTR");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(218+200,75,"ECCF");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(268+200,75,"AM 3");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(318+200,75,"DIM");
floodfill(251,51,YELLOW);

setfillstyle(SOLID_FILL,BLACK);
outtextxy(368+200,75,"DLDA");
floodfill(251,51,YELLOW);
}
```