DESIGN

```
//format of node in linked list
struct node{
        char u_name[];
        int priority;
        struct node * next;
};

//format of link list
struct List{
        node * head;
};

CreateNode(){
        using malloc create a node and return its address.
}

addPassenger(Linked list L, newnode){
        if L->head==NULL:
                L->head=newnode
        else if(L->head->priority  <  newnode->priority)
                make newnode to point to L->head->next;
                make L->head point to new node;
        else
                declare pointers prev , curr;
                make them both point to first node of list;
                while(curr is not null and curr->priority <= newnode->priority){
                        prev=curr;
                        curr=curr->next;
                }
                newnode->next=prev->next
                prev->next=newnode
}

displayReq(Link list L){
        if L->head==NULL
                print -1
        else
                pointer curr=L->head;
                while(curr is not NULL){
                        print curr->u_name and curr->priority
                        curr=curr->next
                }
}

findPriority(Link List L){
        if L->head==NULL
                print -1
                return NULL
        else
                return L->head
}

updatePriority(Link List L, char name[], int np){
        if(L is empty)
                print -1
                return

        pointer curr=first node of list
        if strcmp(curr->u_name, name)==0:
                curr->priority=np
                L->head=curr->next
                addPassenger(L,curr)
                return

        else
                declare pointer prev and curr
                curr=L->head
                while(curr is not NULL)
                        if strcmp(curr->u_name, name)==0:
                                prev->next=curr->next
                                curr->next=NULL
                                curr->priority=np
                                addPassenger(curr)
                                return //escape from this function as we have found the node
                        else
                                prev=curr
                                curr=curr->next
        printf(N) //if it reached till here means no matching node found
}

bookTicket(Link list L){
        if(L->head==NULL)
                print -1

        else
```

```
                print L->head->u_name and L->head->priority
                L->head=L->head->next
}

main(){

        struct Link List L;

        while(True):
                switch(ch):

                if ch=='s':
                        break
                else if ch=='a':
                        newnode=CreateNode()
                        newnode->u_name=input from user
                        newnode->priority=input from user
                        addPassenger(L,newnode)
                else if ch=='d':
                        displayReq(L)
                else if ch=='b':
                        bookTicket(L)
                else if ch=='f':
                        if L->head==NULL
                                print -1
                        else
                                print L->head->u_name and L->head->priority
                else if ch=='u':
                        take name and np as input from user
                        updatePriorty(L,name,np)
}
```