Advanced Data Structure

Arjun. A. S
1BM18CS019

Lab-2 Write-up

Program : Skip list

```
class node
{
public: int data;
        node ** forwd;
        node (int level, int &data)
        {
            forwd = new node * [level+1];
            memset (forwd, 0, sizeof (node *) * (level+1));
            this -> data = data;
        }
        ~node ()
        {
            delete [] forwd;
        }
};
class skiplist
{ public: node *head;
          int data;
          int level;
          int MAX_LEVEL = 6;
          float P
          skiplist ()
          {
              header = new node (MAX_LEVEL, data);
              level = 0
          }
          ~skiplist ()
          {
              delete head;
          }
          void displaylist ();
          bool searchlist (int &);
          void insertlist (int &);
          void deletelist (int &);
};
```

Arjun. A. S

Aayush. K. S
1BM18CS019

```
// insertion of element into Skip list
void skiplist :: insertlist (int &data)
{
    node *x = head
    node *update [MAX_LEVEL + 1];
    memset (update, 0, sizeof (node *) * (MAX_LEVEL+1));
    for (int i= level; i>=0; i--)
    {
        while (x->forwd[i] != NULL &&
                x->forwd[i]->data < data)
            { x= x->forwd[i];
            }
        update[i] =x;
    }
    x = x->forwd[0];
    if (x== NULL || x->data != data)
    {
        int lvl = randomlevel();
        if (lvl >level)
        {
            for (int i = level +1; i <= lvl; i++)
            { update [i] = head;
            }
            level = lvl;
        }
        x= newnode (lvl, data);
        for (int i=0; i<= lvl ; i++)
        { x->forwd[i] = update[i]->forwd[i];
            update [i] -> forwd[i] = x;
        }
    }
}
```

Arjun. A. S
1BM18CS019

// deletion of an element from skiplist

```
void skiplist::deletelist(int &data)
{   node *x = head;
    node *update[MAX_LEVEL+1];
    memset(update, 0, sizeof(node*)*(MAXLEVEL+1));
    for(int i = level; i>=0; i--)
    {   while(x->forwd[i]!=NULL && x->forwd[i]->data
                                    < data)
        {       x = x->forwd[i];
        }
        update[i] = x;
    }
    if(x->value == value)
    {   for(int i=0; i<=level; i++)
        {   if(update[i]->forwd[i]!=x)
                break;
            update[i]->forwd[i] = x->forwd[i];
        }
        delete x;
        while(level>0 && head->forwd[level]==NULL)
            level--;
    }
}
```

// searching an element

```
bool skiplist::searchlist(int &data)
{       node *x=head;
        for(int i=level; i>=0; i--)
        {   while(x->forwd[i]!=NULL &&
                    x->forwd[i]->value < value)
                x = x->forwd[i];
        }
    x = x->forwd[0];
    return x!=NULL && x->value == value;
}
```

Arjun. A. S