```
#include<stdio.h>
#include<curses.h>
#include<iostream>
#include<string.h>
using namespace std;
struct BpTreeNode
{
   char *data;
   BpTreeNode **child_ptr;
   bool leaf;
   int n;
};
struct BpTreeNode *root = NULL, *node_pointer = NULL, *x = NULL;
BpTreeNode * init()
{
   int i;
   node_pointer = new BpTreeNode;
   node_pointer->data = new char[4];
   node_pointer->child_ptr = new BpTreeNode *[5];
   node_pointer->leaf = true;
   node_pointer->n = 0;
   for (i = 0; i < 5; i++)
   {
      node_pointer->child_ptr[i] = NULL;
   }
   return node_pointer;
}

void print( BpTreeNode  *p)
{
   //cout<<endl;
   int i;
   for (i = 0; i < p->n; i++)
   {
      if (p->leaf == false)
      {
         print(p->child_ptr[i]);
      }
      cout << " " << p->data[i];
   }
   if (p->leaf == false)
   {
      print(p->child_ptr[i]);
   }
   //cout<<endl;
}
void sort(char *p, int n)
{
   int i, j, temp;
   for (i = 0; i < n; i++)
   {
      for (j = i; j <= n; j++)
```

```
        {
            if (p[i] > p[j])
            {
                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
}

int split_child( BpTreeNode  *x, int i)
{
    int j, mid;
    BpTreeNode *node_p1, *node_p3, *y;
    node_p3 = init();
    node_p3->leaf = true;
    if (i == -1)
    {
        mid = x->data[2];
        x->data[2] = 0;
        x->n--;
        node_p1 = init();
        node_p1->leaf = false;
        x->leaf = true;
        for (j = 3; j < 4; j++)
        {
            node_p3->data[j - 3] = x->data[j];
            node_p3->child_ptr[j - 3] = x->child_ptr[j];
            node_p3->n++;
            x->data[j] = 0;
            x->n--;
        }
        for(j = 0; j < 5; j++)
        {
            x->child_ptr[j] = NULL;
        }
        node_p1->data[0] = mid;
        node_p1->child_ptr[node_p1->n] = x;
        node_p1->child_ptr[node_p1->n + 1] = node_p3;
        node_p1->n++;
        root = node_p1;
    }
    else
    {
        y = x->child_ptr[i];
        mid = y->data[2];
        y->data[2] = 0;
        y->n--;
        for (j = 3; j < 4; j++)
        {
            node_p3->data[j - 3] = y->data[j];
```

```
            node_p3->n++;
            y->data[j] = 0;
            y->n--;
        }
        x->child_ptr[i + 1] = y;
        x->child_ptr[i + 1] = node_p3;
    }
    return mid;
}
void insert(char a)
{
    int i, temp;
    x = root;
    if (x == NULL)
    {
        root = init();
        x = root;
    }
    else
    {
        if (x->leaf == true && x->n == 4)
        {
            temp = split_child(x, -1);
            x = root;
            for (i = 0; i < (x->n); i++)
            {
                if ((a > x->data[i]) && (a < x->data[i + 1]))
                {
                    i++;
                    break;
                }
                else if (a < x->data[0])
                {
                    break;
                }
                else
                {
                    continue;
                }
            }
            x = x->child_ptr[i];
        }
        else
        {
            while (x->leaf == false)
            {
            for (i = 0; i < (x->n); i++)
            {
                if ((a > x->data[i]) && (a < x->data[i + 1]))
                {
                    i++;
                    break;
```

```
            }
            else if (a < x->data[0])
            {
               break;
            }
            else
            {
               continue;
            }
         }
            if ((x->child_ptr[i])->n == 4)
            {
               temp = split_child(x, i);
               x->data[x->n] = temp;
               x->n++;
               continue;
            }
            else
            {
               x = x->child_ptr[i];
            }
         }
      }
   }
   x->data[x->n] = a;
   sort(x->data, x->n);
   x->n++;
}

int search( BpTreeNode  *p,char k)
{
   //cout<<endl;
   int i;
   for (i = 0; i < p->n; i++)
   {
      if (p->leaf == false)
      {
         search(p->child_ptr[i],k);
      }
      if(p->data[i]==k)
         return 1;
      if(p->data[i]!=k && i==4)
         return 0;
      //cout << " " << p->data[i];
   }
   if (p->leaf == false)
   {
      search(p->child_ptr[i],k);
   }
   //cout<<endl;
   //eturn 0;
```

```
}

int main()
{
   int i, n,s;
   char t,num;
   cout<<"enter the no of charcters to be inserted\n";
   cin>>n;
   for(i = 0; i < n; i++)
   {
      cout<<"enter the alphabet : ";
      cin>>t;
      insert(t);//insering into the tree
      //cout<<endl;
   }
   cout<<"Traversal of constructed tree :\n";
   print(root);//print all the records currently present in the tree.
   //getch();
   cout<<endl;
   cout<<"1.SEARCH \n2. PRINT\n3. EXIT\n";

   while(TRUE){
      cout<<"ENTER YOUR CHOICE:\n";
      cin>>s;
      switch(s){
         case 1: cout<<"Enter element to search:\n";
            cin>>num; //alphabet to search
            //searching a alphabet, if found return 'YES' else 'NO'
            if(search(root,num)){
               cout<<"YES\n";
            }
            if(!search(root,num)){
               cout<<"NO\n";
            }
            break;
         case 2: cout<<"Traversal of B+ tree : ";
            print(root);
            cout<<endl;
         break;

         case 3: exit(0);
      }
   }
return 0;
}
```