#include <iostream>

#include <stdlib.h>

#include <malloc.h>

using namespace std;

const int MAX = 4 ;

const int MIN = 2 ;

struct btnode

{

int count ;

int value[MAX + 1] ;

btnode \*child[MAX + 1] ;

} ;

class btree

{

private :

btnode \*root ;

public :

btree( ) ;

void insert ( int val ) ;

int setval ( int val, btnode \*n, int \*p, btnode \*\*c ) ;

void fillnode ( int val, btnode \*c, btnode \*n, int k ) ;

void split ( int val, btnode \*c, btnode \*n,

int k, int \*y, btnode \*\*newnode ) ;

void del ( int val ) ;

void clear ( btnode \*root, int k ) ;

void show( ) ;

static void display ( btnode \*root ) ;

static void deltree ( btnode \*root ) ;

~btree( ) ;

} ;

btree :: btree( )

{

root = NULL ;

}

void btree :: insert ( int val )

{

int i ;

btnode \*c, \*n ;

int flag ;

flag = setval ( val, root, &i, &c ) ;

if ( flag )

{

n = new btnode ;

n -> count = 1 ;

n -> value[1] = i ;

n -> child[0] = root ;

n -> child[1] = c ;

root = n ;

}

}

int btree :: setval ( int val, btnode \*n, int \*p, btnode \*\*c )

{

int k ;

if ( n == NULL )

{

\*p = val ;

\*c = NULL ;

return 1 ;

}

else

{

if ( setval ( val, n -> child[k], p, c ) )

{

if ( n -> count < MAX )

{

fillnode ( \*p, \*c, n, k ) ;

return 0 ;

}

else

{

split ( \*p, \*c, n, k, p, c ) ;

return 1 ;

}

}

return 0 ;

}

}

void btree :: fillnode ( int val, btnode \*c, btnode \*n, int k )

{

int i ;

for ( i = n -> count ; i > k ; i-- )

{

n -> value[i + 1] = n -> value[i] ;

n -> child[i + 1] = n -> child[i] ;

}

n -> value[k + 1] = val ;

n -> child[k + 1] = c ;

n -> count++ ;

}

void btree :: split ( int val, btnode \*c, btnode \*n,

int k, int \*y, btnode \*\*newnode )

{

int i, mid ;

if ( k <= MIN )

mid = MIN ;

else

mid = MIN + 1 ;

\*newnode = new btnode ;

for ( i = mid + 1 ; i <= MAX ; i++ )

{

( \*newnode ) -> value[i - mid] = n -> value[i] ;

( \*newnode ) -> child[i - mid] = n -> child[i] ;

}

( \*newnode ) -> count = MAX - mid ;

n -> count = mid ;

if ( k <= MIN )

fillnode ( val, c, n, k ) ;

else

fillnode ( val, c, \*newnode, k - mid ) ;

\*y = n -> value[n -> count] ;

( \*newnode ) -> child[0] = n -> child[n -> count] ;

n -> count-- ;

}

void btree :: del ( int val )

{

btnode \* temp ;

if ( ! delhelp ( val, root ) )

cout << endl << "Value " << val << " not found." ;

else

{

if ( root -> count == 0 )

{

temp = root ;

root = root -> child[0] ;

delete temp ;

}

}

}

void btree :: clear ( btnode \*root, int k )

{

int i ;

for ( i = k + 1 ; i <= root -> count ; i++ )

{

root -> value[i - 1] = root -> value[i] ;

root -> child[i - 1] = root -> child[i] ;

}

root -> count-- ;

}

void btree::show()

{

display (root);

}

void btree::display(btnode \*root)

{

int j;

if ( root != NULL )

{

for ( j = 0 ; j < root -> count ; j++ )

{

display ( root -> child[j] ) ;

cout << root -> value[j + 1] << "\t" ;

}

display ( root -> child[j] ) ;

}

}

void btree :: deltree ( btnode \*root )

{

int i;

if ( root != NULL )

{

for (i = 0 ; i < root -> count ; i++ )

{

deltree ( root -> child[i] ) ;

delete ( root -> child[i] ) ;

}

deltree ( root -> child[i] ) ;

delete ( root -> child[i] ) ;

}

}

btree :: ~btree( )

{

{

deltree ( root ) ;

}

int main( )

{

btree b;

int key;

int k;

int choice;

do

{

cout<<"\nChoose option:\n ";

cout<<"1.Insert\n";

cout<<"2.Delete\n";

cout<<"3.Search\n";

cout<<"4.Display\n";

cout<<"5.Quit\n";

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the key : ";

cin>>key;

b.insert(key);

break;

case 2:

cout<<"Enter the key : ";

cin>>key;

b.del(key);

break;

case 3:

cout<<"Btree is :\n";

b.show();

break;

case 4:

exit(1);

default:

cout<<"Wrong choice\n";

break;

}/\*End of switch\*/

}while(choice<6&&choice>0);/\*End of do while\*/

return 0;

}/\*End of main()\*/