```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct nd
{
      int choice;
      struct nd *words[27];
}node;
node *root;
int index(const char *c)
{
      int i;
      if(*c==' ')
      {
            i=26;
      }
      else
      {
            i=(int)*c-97;
      }
      return i;
}
node* createnode(node *r)
{
      int i;
    r=(node*)malloc(sizeof(node));
    if(r)
    {
      r->choice=1;
      for(i=0;i<27;i++)
      r->words[i]=NULL;
      }
      return r;
}
node* create(node *r1,const char *key)
{

      node *r;
      int l=strlen(key);
      int n,i;
      r=r1;
      for(i=0;i<l;i++)
      {
            n=index(key+i);
            if(r->words[n]==NULL)
            r->words[n]=createnode(r->words[n]);
            r=r->words[n];
      }
      r->choice=0;
      return r1;
}
void print(node *r1,const char *c,int x)
{
      int k,i,j,p,n=0,l=strlen(c);
      char a[20];
                for(k=0;k<l;k++)
          {
            a[k]=*(c+k);
            }
      for(i=0;i<27;i++)
      {
            if(r1->words[i]!=NULL)
            {
                  if(n>0 || x==1)
```

```
                  {
                        printf(", ");
                        j=0;
                        while(j!=l)
                        {
                              printf("%c",a[j]);
                              j++;
                        }
                  }
                  x=0;
                  if(i==26) { a[k]=' '; printf(" "); }
                  else { a[k]=(char)(97+i); printf("%c",97+i); }
                  if(r1->words[i]->choice==0)
                  {
                        x=1;
                  }
                  print(r1->words[i],a,x);
                  n++;
            }
      }
}
void predict(const char *c)
{
      node *r;
      int l=strlen(c);
      int n,i,flag=1,x;
      r=root;
      for(i=0;i<l;i++)
      {
            n=index(c+i);
            r=r->words[n];
            flag=1;
            if(r!=NULL)
            {
                  flag=0;
            }
            else
            {
                  printf("not available");
                  break;
            }
      }
      if(flag==0)
      {
            printf("The predicted words (is/are):");
            for(i=0;i<l;i++)
            {
                  printf("%c",*(c+i));
            }
            x=0;
            if(r->choice==0) x=1;
            print(r,c,x);
      }
}
int main()
{
      int i=0,n;
      char
*word[]={"arnab","a","abandon","abandoned","ability","able","about","above","abr
oad","absence","absent","absolute","absolutely","absorb","abuse","abuse","academ
ic","accent","accept","acceptable","access","accident","accidental","accidentall
y","accommodation","accompany","according to","account","account
for","accurate","accurately","accuse","achieve","achievement","acid","acknowledg
e","a couple
```

```
of","acquire","across","act","action","active","actively","activity","actor","ac
tress","actual","actually","a","adapt","add","addition","additional","add
on","address","add up","add up
to","adequate","adequately","adjust","admiration","admire","admit","adopt","adul
t"};
      char cha[10];
      root=createnode(root);
      n=sizeof(word)/sizeof(word[0]);
      while(i!=n)
      {
            root=create(root,word[i]);
            i++;
      }
            printf("Enter 1st one or more alphabet to predict your word:");
      gets(cha);
      predict(cha);
      return 0;
}
```