Procedure fib

```
#include <stdio.h>
#include <stdlib.h>
int n, k, p, b[100],s[100], nb=1;

void type()
{ int i;
  printf("%d   ",nb++);
  for (i=1; i<=n; i++) printf("%d",b[i]);
       printf("\n");
}

void fib(int m, int j)
{ int i;
if( ! (j==0 || m==0 || m==2*j-1) )
{
     if(b[m]==0)
      { fib(m-1,j);
        b[m]=1; b[s[j]]=0;
        s[j]=m;
        type();
        fib(m-2,j-1);
      }
    else
      { fib(m-2,j-1);
        s[j]=s[j-1]+2;
        b[s[j-1]+2]=1;b[m]=0;
        type();
        fib(m-1,j);
      }
}
}

int main()
{ int i;
  printf(" ENTER n: "); scanf("%d", &n);
  printf(" ENTER k: "); scanf("%d", &k);


  if(n<2*k-1) printf("%d>%d",k,n);
  else { for(i=1;i<=n;i++) b[i]=0;
       for(i=0;i<=k-1;i++) b[2*i+1]=1;
```

```
       for(i=1;i<=k;i++) s[i]=2*i-1; s[0]=-1;
       printf("%d   ",nb++);
       for (i=1; i<=n; i++) printf("%d",b[i]);
       printf("\n");
       fib(n, k);
      }
}
```

Procedure pref

```
#include <stdio.h>
int n, k, p, b[100],s[100], nb=1;
#define MAX(a,b) (((a)>(b))?(a):(b))

void type(int u, int v)
{ int i;
  printf("%d   ",nb++);
  for (i=1; i<=n; i++) printf("%d",b[i]);
  printf(",%d %d<- transposed bits",u,v);
       printf("\n");
}

void pref(int m, int j)
{ int u;
 if ( m!=j && j>0)
{   if(b[m]==0)
     { pref(m-1,j);
       u=s[j]; b[m]=1; b[s[j]]=0; s[j]=m;
       type(m,u);
       pref(m-1,j-1);
     }
   else
     { pref(m-1,j-1);
       s[j]=MAX(s[j-1]+1,(p+1)*j);
       if(s[j]!=m)
        { b[m]=0; ; b[s[j]]=1;
          type(m,s[j]);
          pref(m-1,j);
        }
     }
}
}

int main()
{ int i,j=1;
```

```
  printf(" ENTER n: "); scanf("%d", &n);
  printf(" ENTER k: "); scanf("%d", &k);
  printf(" ENTER p (for p=0 combinations are produced, ..): "); scanf("%d", &p);

  if(n<(p+1)*k) printf("ERROR");
  else { for(i=1;i<=n;i++) b[i]=0;
       for(i=1;i<=n;i++)
       {   printf(" ENTER b[%d] ",i); scanf("%d", &b[i]);
           if(b[i]==1){ s[j]=i; j++; }
       }
       printf("%d   ",nb++);
       for (i=1; i<=n; i++) printf("%d",b[i]);printf(",");
       printf("\n");
       pref(n, k);
      }
}
```