

CP 111 Computer Programming

2/2554 Section B01/B02

Lab 10

1. Problem name: Tower_of_Hanoi

```
#include <stdio.h>

void Move(int n,int startPost,int endPost,int tempPost);

int main()
{
    int n,startP,endP,tempP;
    scanf("%d %d %d %d",&n,&startP,&endP,&tempP);
    Move(n,startP,endP,tempP);
    return 0;
}

void Move(int n,int startPost,int endPost,int tempPost)
{
    if(n>1)
    {
        Move(n-1,startPost,tempPost,endPost);
        printf("%d %d %d\n",n,startPost,endPost);
        Move(n-1,tempPost,endPost,startPost);
    }
    else // n=1
    {
        printf("%d %d %d\n",n,startPost,endPost);
    }
}
```

2. Problem name: Recur_Binary_Search

```
#include <stdio.h>

int BinarySearch(int b, int *A, int start, int end);

int main()
{
    int data[100], i, n, b, retval;

    scanf("%d", &n);
    for(i=0; i<n; i++)
        scanf("%d", &data[i]);
    scanf("%d", &b);
    retval=BinarySearch(b, data, 0, n-1);

    printf("%d\n", retval);

    return 0;
}

int BinarySearch(int b, int *A, int start, int end)
{
    int middle;

    middle=(start+end)/2;
    if(start>end)
        return -1;
    else if(A[middle]==b)
        return middle;
    else if(A[middle]>b)
        return (BinarySearch(b, A, middle+1, end));
    else
        return (BinarySearch(b, A, start, middle-1));
}
```

3. Problem name: Iter_Binary_Search

```
#include <stdio.h>

int Iter_Binary_Search(int *A,int n, int b);

int main()
{
    int data[100],i,n,b,retval;

    scanf("%d",&n);
    for(i=0;i<n;i++)
        scanf("%d",&data[i]);
    scanf("%d",&b);
    retval=Iter_Binary_Search(data,n,b);

    printf("%d\n",retval);

    return 0;
}

int Iter_Binary_Search(int *A,int n, int b)
{
    int start,mid,end;

    start=0;
    end=n-1;
    while(1)
    {
        if(start>end)
            return(-1);
        mid=(end+start)/2;
        if(A[mid]==b)
        {
            return(mid);
        }
        else if(A[mid]>b)
        {
            start=mid+1;
            end=end;
        }
        else
        {
            start=start;
            end=mid-1;
        }
    }
}
```

4. Problem name: Recur_Occurrences

```
#include <stdio.h>
int Occurrences(char *,char);
void main()
{
    char buff[100];
    char ch;
    int count;

    gets(buff);
    scanf("%c",&ch);
    count=Occurrences(buff,ch);
    printf("%d\n",count);
    printf("\n\n");
}
int Occurrences(char *str,char ch)
{
    if(*str==0)
        return 0;
    else if(*str==ch)
        return(1+Occurrences(str+1,ch));
    else
        return(Occurrences(str+1,ch));
}
```

5. Problem name: Recur_Minimum

```
#include <stdio.h>
int Recur_Minimum(int *A, int start, int end);
int main()
{
    int data[100],i,n,b,min;
    scanf("%d",&n);
    for(i=0;i<n;i++)
        scanf("%d",&data[i]);
    min=Recur_Minimum(data,0,n-1);
    printf("%d\n",min);
    return 0;
}

int Recur_Minimum(int *A, int start, int end)
{
    int x;
    if(start==end)
        return(A[start]);
    else{
        x=Recur_Minimum(A,start,end-1);
        if(x<A[end])
            return x;
        else
            return A[end];
    }
}
```