SEARCHING AND SORTING

Insertion Sort - Part 1

```
#include <stdio.h>
void print(int ar_size, int* ar) {
   int i;
   for(i=0; i<ar_size; i++) {
      printf("%d ", ar[i]);
   }
   printf("\n");
}

#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <assert.h>

/* Head ends here */
void insertionSort(int ar_size, int *  ar) {
   int j = ar_size-1;
   int v = ar[j];
   while(v < ar[j-1]) {
      ar[j] = ar[j-1];
      j--;
      print(ar_size, ar);
   }
   ar[j] = v;
```

```
    print(ar_size, ar);
}

/* Tail starts here */
int main() {

   int _ar_size;
scanf("%d", &_ar_size);
int _ar[_ar_size], _ar_i;
for(_ar_i = 0; _ar_i < _ar_size; _ar_i++) {
   scanf("%d", &_ar[_ar_i]);
}

insertionSort(_ar_size, _ar);

   return 0;
}
```

Insertion Sort - Part 2

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <assert.h>
/* Head ends here */
void insertionSort(int ar_size, int *  ar) {
  for (int i = 1; i < ar_size; ++i) {
    int j = i - 1;
    int p = ar[i];
    while (j >= 0 && p < ar[j]) {
      ar[j+1] = ar[j];
      j--;
    }
    ar[j+1] = p;
    printf("%d", ar[0]);
    for (int k = 1; k < ar_size; ++k) {
      printf(" %d", ar[k]);
    }
    printf("\n");
  }
}
/* Tail starts here */
int main() {
  int _ar_size;
scanf("%d", &_ar_size);
int _ar[_ar_size], _ar_i;
for(_ar_i = 0; _ar_i < _ar_size; _ar_i++) {
  scanf("%d", &_ar[_ar_i]);
}

insertionSort(_ar_size, _ar);

  return 0;
}
```

Correctness and the Loop Invariant

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <assert.h>
/* Head ends here */
#include <stddef.h>
void insertionSort(int ar_size, int *  ar) {
   int i,j;
   int value;
   for(i=1;i<ar_size;i++)
   {
      value=ar[i];
      j=i-1;
      while(j>=0 && value<ar[j])
      {
         ar[j+1]=ar[j];
         j=j-1;
      }
      ar[j+1]=value;
   }
   for(j=0;j<ar_size;j++)
      {
         printf("%d",ar[j]);
         printf(" ");
      }
}
/* Tail starts here */
int main(void) {
  int _ar_size;
scanf("%d", &_ar_size);
int _ar[_ar_size], _ar_i;
for(_ar_i = 0; _ar_i < _ar_size; _ar_i++) {
  scanf("%d", &_ar[_ar_i]);
}

insertionSort(_ar_size, _ar);

  return 0;
}
```

Running Time of Algorithms

```c
#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include <assert.h>


/* Head ends here */

void insertionSort(int ar_size, int *  ar,int *shifts) {

int temp=ar[ar_size-1],i;

  for(i=ar_size-2;i>=0;i--)

  {

      if(ar[i]>temp){

           ar[i+1]=ar[i];

           *shifts=*shifts+1;

           }

      else

           break;



  }

  ar[i+1]=temp;





}


/* Tail starts here */
```

```
int main() {

  int _ar_size,i,j,shifts=0;
scanf("%d", &_ar_size);
int _ar[_ar_size], _ar_i;
for(_ar_i = 0; _ar_i < _ar_size; _ar_i++) {
  scanf("%d", &_ar[_ar_i]);
}
for(i=2;i<=_ar_size;i++)
{
   insertionSort(i, _ar,&shifts);
}
   printf("%d",shifts);
  return 0;
}
```

Counting Sort 1

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

   int n,i;
   int b[100],a;
```

```
    scanf("%d",&n);

    for(i=0;i<100;i++)
    {
        b[i]=0;
    }
    for(i=0;i<n;i++)
    {
        scanf("%d",&a);
        b[a]++;
    }
    for(i=0;i<100;i++)
    {
        printf("%d ", b[i]);
    }

    return 0;
}
```

RECURSION AND BIT MANIPULATION

Crossword Puzzle

The Power Sum

```
#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>


int the_power_sum(int n, int m,int p){

   int tmp = pow(m,p);

   if(tmp == n) return 1;

   if(tmp > n) return 0;

 return the_power_sum(n,m+1,p) + the_power_sum(n-tmp, m+1,p);

}

int main() {

   int n,p;

   scanf("%d%d",&n,&p);

   printf("%d", the_power_sum(n,1,p));


   return 0;

}
```

Counter Game

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
int isPow2(long unsigned  int);
unsigned long int largePow(long unsigned int);
int main() {

    int t,i,win;
   long unsigned int n;
    scanf("%d",&t);
    for(i=0;i<t;++i)
        {
        win=0;
        scanf("%lu",&n);
        if(n==1)
            printf("Richard\n");
        else
            {
            while(n!=1)
                {
                if(isPow2(n))
                    n>>=1;
                else
                    n-=largePow(n);
                ++win;
```

```
      }
    }
    if(win%2==0)
      printf("Richard\n");
    else
      printf("Louise\n");
  }
  return 0;
}
int isPow2(long unsigned int n)
  {
  return !(n&(n-1));
}
long unsigned int largePow(long unsigned int n)
  {
  long unsigned int m;
  while(n)
    {
    m=n;
    n=n&(n-1);
  }
  return m;
}
```

GREEDY AND DYNAMIC PROGRAMMING

The Coin Change Problem

Sherlock and Cost

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main() {
  int T,N,B,L,R,ML,MR,X,Y,P,Q;
  scanf("%d",&T);
  for(int i = 0; i < T; i++) {
    scanf("%d",&N);
    for(int j = 0; j < N; j++) {
      scanf("%d",&B);
      if(j) {
        X = L - 1 + ML;
        Y = R - 1 + MR;
        P = abs(L - B) + ML;
        Q = abs(R - B) + MR;
        ML = (X > Y ? X : Y);
        MR = (P > Q ? P : Q);
      } else {
        ML = MR = 0;
      }
      L = 1;
      R = B;
    }
    printf("%d\n", (ML > MR ? ML : MR));
  }
  return 0;
}
```