HW12:

interface:

element\* CreateMinHeap(int\* top);

void Insert(element\* target\_heap,int\* top);

int RemoveMinHeap(element\* target\_heap,int \*top);

void ChangePrior(element\* target\_heap,int\*top);

第一步: 利用CreateMinHeap初始化Heap 陣列

element\* heap;

**int** top;

    heap=CreateMinHeap(&top);

之後，便能操作另外三種函式:

void Insert(element\* target\_heap,int\* top);

int RemoveMinHeap(element\* target\_heap,int \*top);

void ChangePrior(element\* target\_heap,int\*top);

這些函式被呼叫時，會同時要求使用者輸入欲插入的數值，因此使用上不用於程式碼中宣告變數，並一再讀入,在使用上只需要知道欲插入多少數值即可以for loop撰寫。

程式碼: CreateMinHeap

element\* CreateMinHeap(**int**\* top){

**static** element minheap[MAX\_SIZE];*//must use static variable when returning pointer*

    \*top=1;

    return &minheap[0];

};

說明: 這個函式是用來初始化heap ，將宣告的空間以指標回傳。

程式碼: Insert

**int** i=\*top;

    (target\_heap+(i))->key=insert\_key;

**int** temp;

    for(;i>1;){

        if(i%2==1){*//insert to the right child*

            if((target\_heap+i)->key<(target\_heap+(i-1)/2)->key)

            {

                temp=(target\_heap+i)->key;

                (target\_heap+i)->key=(target\_heap+(i-1)/2)->key;

                (target\_heap+(i-1)/2)->key=temp;

                i=(i-1)/2;

            }else{break;}

       }else

        if(i%2==0){*//insert to the left child*

            if((target\_heap+i)->key<(target\_heap+i/2)->key)

            {

                temp=(target\_heap+i)->key;

                (target\_heap+i)->key=(target\_heap+i/2)->key;

                (target\_heap+i/2)->key=temp;

                i/=2;

            }else{break;}

        }

    }

    ++\*top;

說明: 將新的數值放在Heap array的最上層之後，進行向上(bottom-up)的整理，如果新的值小於parent，則兩者交換，直到最頂端(前端)。

程式碼:RemoveMinHeap

**int** RemoveMinHeap(element\* target\_heap,**int** \*top){

**int** temp,min\_result=(target\_heap+1)->key;

    (target\_heap+1)->key=(target\_heap+\*top-1)->key;

    for(**int** i=1;i<\*top;){

        if((target\_heap+i\*2)->key==0){

*//left child==0( suggest right child ==0 )*

            break;

        }else if((target\_heap+i\*2)->key!=0 &&(target\_heap+i\*2+1)->key==0){

*// left child !=0 and right child==0*

            (target\_heap+i)->key=(target\_heap+i\*2)->key;

            i=2\*i;

        }

        else if((target\_heap+i\*2)->key!=0 &&(target\_heap+i\*2+1)->key!=0){

*//right and left both !=0*

            if( (target\_heap+i\*2)->key<(target\_heap+i\*2+1)->key){

*//when left child is smaller*

                if((target\_heap+i\*2)->key < (target\_heap+i)->key){

*//and the parent is larger than left*

*//swape parent and left child*

                    temp=(target\_heap+i)->key;

                    (target\_heap+i)->key=(target\_heap+i\*2)->key;

                    (target\_heap+i\*2)->key=temp;

                    i=i\*2;

                }*//if left child is smaller and the parent is smaller than left child,*

                else break;*//do nothing*

            }else if( (target\_heap+i\*2)->key>= (target\_heap+i\*2+1)->key){

*//when right child is smaller*

                if((target\_heap+i\*2+1)->key < (target\_heap+i)->key){

*//and the parent is largert than right*

*//swape parent and right child*

                    temp=(target\_heap+i)->key;

                    (target\_heap+i)->key=(target\_heap+i\*2+1)->key;

                    (target\_heap+i\*2+1)->key=temp;

                    i=i\*2+1;

                }else break;

            }

        }

    }

    --\*top;

    (target\_heap+\*top)->key=0;

    return min\_result;

};

說明: 先將陣列中最前端與最尾端兩個數值交換，之後進行向下(top-down)的方式與left child, right child 進行比較，最小的會與原parent交換成為新parent，並在交換後重複向下執行直到array最尾端。

程式碼: ChangePrior

**void** ChangePrior(element\* target\_heap,**int**\*top){

**int** change\_key, priority;

    printf("\nenter the num to change priority:");

    scanf("%d",&change\_key);

**int** cur=IsExisted(target\_heap,\*top,change\_key);

    printf("position of change key:%d\n",cur);

    if(cur==0)

    {

        printf("no such num existed\n");

        return;

    }

    printf("enter the priority to change to:");

    scanf("%d",&priority);

    element copy[MAX\_SIZE];

**int** arr[\*top];

**int** counter=\*top;

**int** cp\_top=counter;

    for(**int** i=1;i<counter;i++){

        copy[i]=\*(target\_heap+i);

    }

    for(**int** i=1;i<counter;i++){

        arr[i]=RemoveMinHeap(&copy[0],&cp\_top);

    }

      for(**int** i=1;i<counter;i++){

         printf("arr[i]=%d\n",arr[i]);

    }

**int** prio\_value=arr[priority];

**int** prior\_index;

    for(**int** i=1;i<counter;i++)

        if((target\_heap+i)->key==prio\_value){ prior\_index=i;break;}

**int** temp=(target\_heap+cur)->key;

    (target\_heap+cur)->key=(target\_heap+prior\_index)->key;

    (target\_heap+prior\_index)->key=temp;

};

說明: 在決定欲更改優先權的值與優先權順位後，這個函式會先搜尋欲修改的數值的位置，若堆疊中不存在該值，則結束這個函式。當值存在，這個函式會計算欲修改數值於陣列中的位置，與該優先權順位原本的數值，以及其存在於陣列中的位置。 最後將兩者互換，達到修改優先順序的效果。(然而當執行了insert 或 remove後 順序可能會發生改變)