```
1  void bubbleSort(Pessoas *P, int n){
2     int i, j; Pessoas aux;
3     for (i = 1; i < n; i++) {
4       for (j = n; j > i; j--) {
5         if (P[j].Chave < P[j - 1].Chave){ aux = P[j];P[j] = P[j - 1];P[j - 1]= aux;}}}}
6  void selectionSort(Pessoas *P, int n){
7    int i, j, Min; Pessoas x;
8    for (i = 0; i < n - 1; i++){
9      Min = i;
10     for (j = i + 1; j < n; j++){if (P[j].Chave < P[Min].Chave) Min = j;}
11      x = P[Min];P[Min] = P[i]; P[i] = x; } }
12 void insertionSort(Pessoas *A, int n){
13     int i, j;Pessoas x;
14     for(i=2; i<=n; i++){
15         x = A[i]; j = i - 1;
16         A[0] = x;///Sentinela
17         while(x.Chave<A[j].Chave){ A[j+1] = A[j]; j--;}
18         A[j+1] = x;}
19 ///Metodo Quicksort
20 void particao(int Esq, int Dir, int *i, int *j, Pessoas *A){
21  Pessoas x, aux; *i=Esq;*j=Dir; x=A[(((*i)+(*j))/2)]; ///PIVO
22   do{
23     while(x.Chave>A[*i].Chave) (*i)++;
24     while(x.Chave<A[*j].Chave) (*j)--;
25     if(*i<=*j){ aux=A[*i]; A[*i]=A[*j]; A[*j]=aux;(*i)++;(*j)--;}
26   }while(*i<=*j);}
27 void ordena(int Esq,int Dir, Pessoas *A){
28     int i,j;
29     particao(Esq,Dir,&i,&j,A);
30     if(Esq<j) ordena(Esq,j,A);
31     if(i<Dir) ordena(i,Dir,A);}
32 void quicksort(Pessoas *A, contador n){ordena(1,n,A);}
33 ///Metodo HeapSort
34 void Insere (TipoItem *x , TipoItem *A, TipoIndice *n){
35   (*n)++; A[*n]=*x; A[*n].Chave=INT_MIN;
36   AumentaChave(*n,x->Chave,A);}
37 void AumentaChave(TipoIndice i, TipoChave ChaveNova, TipoItem *A){
38   TipoItem x;
39   if(ChaveNova < A[i].Chave){printf("ChaveNova menor que atual \n"); return;}
40   A[i].Chave=ChaveNova;
41   while(i > 1 && (A[i/2].Chave<A[i].Chave)){x=A[i/2];A[i/2]=A[i];A[i]=x; i=i/2; } }
42 TipoItem RetiraMax(TipoItem *A, TipoIndice *n){
43   TipoItem Maximo;
44   if (*n < 1) printf("Erro: heap vazio \n" );
45   else{Maximo = A[1];A[1] = A[*n];(*n)--;RefazMax(1, *n, A);}
46   return Maximo;}
47 void ConstroiMax(TipoItem *A, TipoIndice n){
48   TipoIndice Esq = n/2 + 1;
49   while (Esq > 1){
50     Esq--;
51     RefazMax(Esq, n, A);}}
52 void RefazMax(TipoIndice Esq, TipoIndice Dir, TipoItem *A){
53   TipoIndice i = Esq; TipoItem x; int j = i*2;
54   x = A[i];
55   while (j<=Dir){
56     if (j<Dir)
57       if (A[j].Chave<A[j+1].Chave)///inverte no Heap minimo
58           j++;
59     if (x.Chave>=A[j].Chave)///Inverte no Heap Minimo
60         break;
61     A[i]=A[j]; i=j; j=i*2;}
62     A[i]=x;}
63 /// Metodo MergeSort
64 void intercala(int * v, int left, int mid, int right){
65     int aux[right-left+1];
66     int i = left, j = mid + 1, k = 0;
67     while(i <= mid && j <= right){
68         if(v[i] <= v[j]) aux[k++] = v[i++];
69         else aux[k++] = v[j++];}
70     while(i <= mid) aux[k++] = v[i++];
71     while(j <= right) aux[k++] = v[j++];
72     for(i = left, k = 0; i <= right; i++, k++) v[i] = aux[k];
73     return;}
74 void mergeaux(int *v, int left, int right){
75     if(left >= right)return;
76     int mid = floor((right + left) / 2);
77     mergeaux(v, left, mid);
78     mergeaux(v, mid+1, right);
79     intercala(v, left, mid, right);
80     return;}
81 void mergesort(int *v, int n, int *p){mergeaux(v, 0, n-1);return;}
82
```