|  |
| --- |
| 第一部分：实验分析与设计   1. **实验目的和要求**   1. 目的  线性表应用  2. 要求  约瑟夫环问题  设计一个一元稀疏多项式简单计算器，要求基本功能：  输入并建立多项式  输出多项式  两个多项式相加  两个多项式相减   1. **分析与设计**   1. 数据结构设计  （1）  struct List{  int arr[N];  int n;  }list;  （2）  struct node{  int expn;  double coef;  node\* next;  node() {  expn = coef = 0;  next = NULL;  }  node(int expn,double coef) :  expn(expn),coef(coef) {  next = NULL;  }  };  2. 核心算法设计  (1)  void del(int pos){  if(pos >= n || pos < 0) return;  for(;pos + 1 < n;pos++){  arr[pos] = arr[pos + 1];  }  n--;  }  (2)  struct poly{  node\* head;  poly(){  head = new node();  }  void insert(node\* nod){  node\* cur = head;  while(cur->next != NULL){  if(cur->next->expn == nod->expn){  cur->next->coef += nod->coef;  return ;  }else if(cur->next->expn > nod->expn){  break;  }  cur = cur->next;  }  nod->next = cur->next;  cur->next = nod;  }  void add(poly\* pol){  node\* last = head;  node\* pos1 = head->next;  node\* pos2 = pol->head->next;  while(pos1 != NULL && pos2 != NULL){  if(pos1->expn == pos2->expn){  pos1->coef += pos2->coef;  last = pos1;  pos1 = pos1->next;  pos2 = pos2->next;  }else if(pos1->expn < pos2->expn){  last = pos1;  pos1 = pos1->next;  }else{  node\* tmp = new node(pos2->expn,pos2->coef);  tmp->next = last->next;  last->next = tmp;  pos2 = pos2->next;  }  }  while(pos2 != NULL){  node\* tmp = new node(pos2->expn,pos2->coef);  last->next = tmp;  pos2 = pos2->next;  }  }  void sub(poly\* pol){  node\* last = head;  node\* pos1 = head->next;  node\* pos2 = pol->head->next;  while(pos1 != NULL && pos2 != NULL){  if(pos1->expn == pos2->expn){  pos1->coef -= pos2->coef;  last = pos1;  pos1 = pos1->next;  pos2 = pos2->next;  }else if(pos1->expn < pos2->expn){  last = pos1;  pos1 = pos1->next;  }else{  node\* tmp = new node(pos2->expn,-pos2->coef);  tmp->next = last->next;  last->next = tmp;  pos2 = pos2->next;  }  }  while(pos2 != NULL){  node\* tmp = new node(pos2->expn,-pos2->coef);  last->next = tmp;  pos2 = pos2->next;  }  }  void print(){  node\* cur = head->next;  while(cur != NULL){  printf("%.2lfX^%d ",cur->coef,cur->expn);  cur = cur->next;  }  printf("\n");  }  void del(){  node\* cur = head->next;  while(cur != NULL){  node\* tmp = cur;  cur = cur->next;  free(tmp);  }  free(head);  }  }p1,p2;  3. 测试用例设计  (1)  void init(){  for(int i = 0;i < n;i++){  arr[i] = i + 1;  }  }  (2)  int main(){  int ex;  double co;  scanf("%d %d",&n,&m);  for(int i = 0;i < n;i++){  scanf("%lf %d",&co,&ex);  node\* tmp = new node(ex,co);  p1.insert(tmp);  }  for(int i = 0;i < m;i++){  scanf("%lf %d",&co,&ex);  node\* tmp = new node(ex,co);  p2.insert(tmp);  }  p1.add(&p2);  p1.print();  p2.print();  p1.del();  p2.del();  return 0;  }  三、**实验软硬件环境**  {  "version": "2.0.0",  "tasks": [  {  "type": "shell",  "label": "C/C++: g++.exe 生成活动文件",  "command": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe",  "args": [  "-g",  "${file}",  "-o",  "${fileDirname}\\${fileBasenameNoExtension}.exe"  ],  "options": {  "cwd": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin"  },  "problemMatcher": [  "$gcc"  ],  "group": {  "kind": "build",  "isDefault": true  },  "detail": "编译器: E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe"  }  ]  }  第二部分：实验过程与结果  **一、实验说明（所建工程文件等的说明）**    **二、调试说明（调试手段、过程及结果分析）**  vscode 断点调试 codeRunner 运行  过程顺利  结果无误   1. **软件测试（测试效果.界面、综合分析和结论）**       第三部分：实验小结、收获与体会  **对于链表的crud操作有了更深入的认识，增强了代码能力，提升了算法思维,对于调试程序的操作更为熟悉，学习了出现问题如何分析原因追溯源头，培养了不骄不躁，埋下头学习的刻苦优秀品质。**  第四部分：附源程序  （1）  #include <cstdio>  #include <cstdlib>  using namespace std;  const int N = 1e5 + 10;  struct List{  int arr[N];  int n;  void init(){  for(int i = 0;i < n;i++){  arr[i] = i + 1;  }  }  void del(int pos){  if(pos >= n || pos < 0) return;  for(;pos + 1 < n;pos++){  arr[pos] = arr[pos + 1];  }  n--;  }  }list;  int m;  int main(){  scanf("%d %d",&list.n,&m);  list.init();  int t = 0;  while(list.n){  t = (t + m - 1) % list.n;  printf("%d ",list.arr[t]);  list.del(t);  }  return 0;  }  (2)  #include <cstdio>  #include <cstdlib>  using namespace std;  struct node{  int expn;  double coef;  node\* next;  node() {  expn = coef = 0;  next = NULL;  }  node(int expn,double coef) :  expn(expn),coef(coef) {  next = NULL;  }  };  struct poly{  node\* head;  poly(){  head = new node();  }  void insert(node\* nod){  node\* cur = head;  while(cur->next != NULL){  if(cur->next->expn == nod->expn){  cur->next->coef += nod->coef;  return ;  }else if(cur->next->expn > nod->expn){  break;  }  cur = cur->next;  }  nod->next = cur->next;  cur->next = nod;  }  void add(poly\* pol){  node\* last = head;  node\* pos1 = head->next;  node\* pos2 = pol->head->next;  while(pos1 != NULL && pos2 != NULL){  if(pos1->expn == pos2->expn){  pos1->coef += pos2->coef;  last = pos1;  pos1 = pos1->next;  pos2 = pos2->next;  }else if(pos1->expn < pos2->expn){  last = pos1;  pos1 = pos1->next;  }else{  node\* tmp = new node(pos2->expn,pos2->coef);  tmp->next = last->next;  last->next = tmp;  pos2 = pos2->next;  }  }  while(pos2 != NULL){  node\* tmp = new node(pos2->expn,pos2->coef);  last->next = tmp;  pos2 = pos2->next;  }  }  void sub(poly\* pol){  node\* last = head;  node\* pos1 = head->next;  node\* pos2 = pol->head->next;  while(pos1 != NULL && pos2 != NULL){  if(pos1->expn == pos2->expn){  pos1->coef -= pos2->coef;  last = pos1;  pos1 = pos1->next;  pos2 = pos2->next;  }else if(pos1->expn < pos2->expn){  last = pos1;  pos1 = pos1->next;  }else{  node\* tmp = new node(pos2->expn,-pos2->coef);  tmp->next = last->next;  last->next = tmp;  pos2 = pos2->next;  }  }  while(pos2 != NULL){  node\* tmp = new node(pos2->expn,-pos2->coef);  last->next = tmp;  pos2 = pos2->next;  }  }  void print(){  node\* cur = head->next;  while(cur != NULL){  printf("%.2lfX^%d ",cur->coef,cur->expn);  cur = cur->next;  }  printf("\n");  }  void del(){  node\* cur = head->next;  while(cur != NULL){  node\* tmp = cur;  cur = cur->next;  free(tmp);  }  free(head);  }  }p1,p2;  int n,m;  int main(){  int ex;  double co;  scanf("%d %d",&n,&m);  for(int i = 0;i < n;i++){  scanf("%lf %d",&co,&ex);  node\* tmp = new node(ex,co);  p1.insert(tmp);  }  for(int i = 0;i < m;i++){  scanf("%lf %d",&co,&ex);  node\* tmp = new node(ex,co);  p2.insert(tmp);  }  p1.add(&p2);  p1.print();  p2.print();  p1.del();  p2.del();  return 0;  } |
| 第一部分：实验分析与设计   1. **实验目的和要求**   1. 目的  树的应用  2. 要求  假设用于通信的电文由字符集{a,b,c,d,e,f,g}中的字母构成，它们在电文中出现的频度分别为{0.31, 0.16, 0.10, 0.08, 0.11, 0.20, 0.04}，设计一个哈夫曼编码器，为这7个字母设计哈夫曼编码。   1. **分析与设计**   1. 数据结构设计  struct tree{  char data;  double freq;  int lson,rson;  tree() : lson(-1),rson(-1) {}  tree(double freq,int lson,int rson) :  freq(freq),lson(lson),rson(rson) {}  }tre[N];  struct node{  int idx;  double freq;  node() {}  node(int idx,double freq) :  idx(idx),freq(freq) {}  bool operator < (const node& ot)const{  return freq > ot.freq;  }  };  2. 核心算法设计  int n,tot;  double arr[N];  string ans[N];  void dfs(int,int,int);  int main(){  double t;  priority\_queue <node> q;  cin >> n;  for(int i = 0;i < n;i++){  cin >> tre[tot].freq;  tre[tot].data = i + 'a';  q.push(node(tot,tre[tot].freq));  tot++;  }  node root;  while(!q.empty()){  root = q.top();  q.pop();  if(q.empty()) break;  node sec = q.top();  q.pop();  tre[tot] = tree(root.freq + sec.freq,root.idx,sec.idx);  q.push(node(tot,tre[tot].freq));  tot++;  }  dfs(root.idx,-1,-1);  for(int i = 0;i < n;i++){  cout << (char)('a' + i) << " : " << ans[i] << endl;  }  return 0;  }  void dfs(int cur,int fa,int sid){  if(fa != -1) ans[cur] = ans[fa] + (sid == 1 ? "1" : "0");  if(tre[cur].lson != -1) dfs(tre[cur].lson,cur,1);  if(tre[cur].rson != -1) dfs(tre[cur].rson,cur,0);  }  3. 测试用例设计  7  0.31 0.16 0.10 0.08 0.11 0.20 0.04  三、**实验软硬件环境**  {  "version": "2.0.0",  "tasks": [  {  "type": "shell",  "label": "C/C++: g++.exe 生成活动文件",  "command": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe",  "args": [  "-g",  "${file}",  "-o",  "${fileDirname}\\${fileBasenameNoExtension}.exe"  ],  "options": {  "cwd": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin"  },  "problemMatcher": [  "$gcc"  ],  "group": {  "kind": "build",  "isDefault": true  },  "detail": "编译器: E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe"  }  ]  }  第二部分：实验过程与结果  **一、实验说明（所建工程文件等的说明）**    **二、调试说明（调试手段、过程及结果分析）**  vscode 断点调试 codeRunner 运行  过程顺利  结果无误   1. **软件测试（测试效果.界面、综合分析和结论）**     第三部分：实验小结、收获与体会  **对于树的操作与应用有了更深入的认识，增强了代码能力，提升了算法思维,对于调试程序的操作更为熟悉，学习了出现问题如何分析原因追溯源头，培养了不骄不躁，埋下头学习的刻苦优秀品质。**  第四部分：附源程序  #include <cstdio>  #include <cstdlib>  #include <iostream>  #include <queue>  #include <string>  using namespace std;  const int N = 50;  struct tree{  char data;  double freq;  int lson,rson;  tree() : lson(-1),rson(-1) {}  tree(double freq,int lson,int rson) :  freq(freq),lson(lson),rson(rson) {}  }tre[N];  struct node{  int idx;  double freq;  node() {}  node(int idx,double freq) :  idx(idx),freq(freq) {}  bool operator < (const node& ot)const{  return freq > ot.freq;  }  };  int n,tot;  double arr[N];  string ans[N];  void dfs(int,int,int);  int main(){  double t;  priority\_queue <node> q;  cin >> n;  for(int i = 0;i < n;i++){  cin >> tre[tot].freq;  tre[tot].data = i + 'a';  q.push(node(tot,tre[tot].freq));  tot++;  }  node root;  while(!q.empty()){  root = q.top();  q.pop();  if(q.empty()) break;  node sec = q.top();  q.pop();  tre[tot] = tree(root.freq + sec.freq,root.idx,sec.idx);  q.push(node(tot,tre[tot].freq));  tot++;  }  dfs(root.idx,-1,-1);  for(int i = 0;i < n;i++){  cout << (char)('a' + i) << " : " << ans[i] << endl;  }  return 0;  }  void dfs(int cur,int fa,int sid){  if(fa != -1) ans[cur] = ans[fa] + (sid == 1 ? "1" : "0");  if(tre[cur].lson != -1) dfs(tre[cur].lson,cur,1);  if(tre[cur].rson != -1) dfs(tre[cur].rson,cur,0);  } |
| 第一部分：实验分析与设计   1. **实验目的和要求**   1. 目的  图的应用  2. 要求  教学计划编制问题。基本要求：   1. 输入参数包括：学期总数，一学期的学分上限，每门课的课程号（固定占3位的字母数字串）、学分和直接先修课的课程号。 2. 允许用户指定下列两种编排策略之一：一是使学生在各学期中的学习负担尽量均匀；二是使课程尽可能地集中在前几个学期中。   若根据给定的条件问题无解，则报告适当的信息；否则将教学计划输出到用户指定的文件中。计划的表格格式自行设计。   1. **分析与设计**   1. 数据结构设计  struct cours{  string id;  int sco;  }cou[N];  struct node{  int idx,sco;  node(int idx,int sco) :  idx(idx),sco(sco) {}  bool operator < (const node& ot) const{  return sco > ot.sco;  }  };  2. 核心算法设计  int term,scoLim,n;  int inc[N],ans[N];  vector <int> path[N];  map <string,int> simap;  int main(){  freopen("out.txt","w",stdout);  string str;  cin >> term >> scoLim >> n;  for(int i = 0;i < n;i++){  cin >> cou[i].id >> cou[i].sco >> str;  simap[cou[i].id] = i;  int pre = -1;  if(str != "-1"){  pre = simap[str];  path[pre].push\_back(i);  inc[i]++;  }  }    int cnt = 0;  priority\_queue <node> q;  for(int i = 0;i < n;i++){  if(inc[i] == 0) q.push(node(i,cou[i].sco));  }  while(!q.empty()){  node cur = q.top();  q.pop();  ans[cnt++] = cur.idx;  for(auto to : path[cur.idx]){  inc[to]--;  if(!inc[to]) q.push(node(to,cou[to].sco));  }  }  bool isLegal = true;  for(int i = 0;i < n && isLegal;i++){  if(inc[i]) isLegal = false;  }  if(!isLegal){  cout << "course conflict!\n";  exit(0);  }  int mode = 0;  cin >> mode;  int curTerm = 2,sum = 0,tot = 0;  cout << "term 1:\n";  for(int i = 0;i < n;i++){  sum += cou[ans[i]].sco;  tot += 1;  if(sum > scoLim || (mode && tot > ceil(n \* 1.0 / term))){  sum = cou[ans[i]].sco;  tot = 1;  cout << "term " << curTerm++ << ":\n";  }  cout << cou[ans[i]].id << endl;  }  while(curTerm <= term){  cout << "term " << curTerm++ << ":\n";  }  fclose(stdout);  return 0;  }  3. 测试用例设计  /\*  4 20 10  a1 2 -1  b2 1 -1  c3 5 a1  d4 7 b2  e5 3 b2  f6 5 d4  g7 6 e5  h8 1 a1  i9 2 -1  j10 3 i9  0  \*/  /\*  4 20 10  a1 2 -1  b2 1 -1  c3 5 a1  d4 7 b2  e5 3 b2  f6 5 d4  g7 6 e5  h8 1 a1  i9 2 -1  j10 3 i9  1  \*/  三、**实验软硬件环境**  {  "version": "2.0.0",  "tasks": [  {  "type": "shell",  "label": "C/C++: g++.exe 生成活动文件",  "command": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe",  "args": [  "-g",  "${file}",  "-o",  "${fileDirname}\\${fileBasenameNoExtension}.exe"  ],  "options": {  "cwd": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin"  },  "problemMatcher": [  "$gcc"  ],  "group": {  "kind": "build",  "isDefault": true  },  "detail": "编译器: E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe"  }  ]  }  第二部分：实验过程与结果  **一、实验说明（所建工程文件等的说明）**    **二、调试说明（调试手段、过程及结果分析）**  vscode 断点调试 codeRunner 运行  过程顺利  结果无误   1. **软件测试（测试效果.界面、综合分析和结论）**       第三部分：实验小结、收获与体会  **对于图的操作和应用有了更深入的认识，增强了代码能力，提升了算法思维,对于调试程序的操作更为熟悉，学习了出现问题如何分析原因追溯源头，培养了不骄不躁，埋下头学习的刻苦优秀品质。**  第四部分：附源程序  #include <cstdio>  #include <cstdlib>  #include <iostream>  #include <queue>  #include <string>  #include <map>  #include <cmath>  using namespace std;  const int N = 50;  struct cours{  string id;  int sco;  }cou[N];  struct node{  int idx,sco;  node(int idx,int sco) :  idx(idx),sco(sco) {}  bool operator < (const node& ot) const{  return sco > ot.sco;  }  };  int term,scoLim,n;  int inc[N],ans[N];  vector <int> path[N];  map <string,int> simap;  int main(){  freopen("out.txt","w",stdout);  string str;  cin >> term >> scoLim >> n;  for(int i = 0;i < n;i++){  cin >> cou[i].id >> cou[i].sco >> str;  simap[cou[i].id] = i;  int pre = -1;  if(str != "-1"){  pre = simap[str];  path[pre].push\_back(i);  inc[i]++;  }  }    int cnt = 0;  priority\_queue <node> q;  for(int i = 0;i < n;i++){  if(inc[i] == 0) q.push(node(i,cou[i].sco));  }  while(!q.empty()){  node cur = q.top();  q.pop();  ans[cnt++] = cur.idx;  for(auto to : path[cur.idx]){  inc[to]--;  if(!inc[to]) q.push(node(to,cou[to].sco));  }  }  bool isLegal = true;  for(int i = 0;i < n && isLegal;i++){  if(inc[i]) isLegal = false;  }  if(!isLegal){  cout << "course conflict!\n";  exit(0);  }  int mode = 0;  cin >> mode;  int curTerm = 2,sum = 0,tot = 0;  cout << "term 1:\n";  for(int i = 0;i < n;i++){  sum += cou[ans[i]].sco;  tot += 1;  if(sum > scoLim || (mode && tot > ceil(n \* 1.0 / term))){  sum = cou[ans[i]].sco;  tot = 1;  cout << "term " << curTerm++ << ":\n";  }  cout << cou[ans[i]].id << endl;  }  while(curTerm <= term){  cout << "term " << curTerm++ << ":\n";  }  fclose(stdout);  return 0;  } |
| 第一部分：实验分析与设计   1. **实验目的和要求**   1. 目的  查找与排序  2. 要求  哈希表设计。针对某个集体（比如你所在的班级）中的“人名”设计一个哈希表，使得平均查找长度不超过R，完成相应的建表和查表程序。  内部排序算法比较。编制一个演示内部排序算法比较的程序。   1. **分析与设计**   1. 数据结构设计  (1)  string store[N];  (2)  int quickArr[N],quickN;  int heapTree[N],heapN;  bool heapUsed[N];  int mergeArr[N],mergeN;  2. 核心算法设计  (1)  int strHash(const string& str){  int res = 1;  for(auto ch: str){  res = (res + (ch - 'A')) % MOD;  }  return res;  }  void pushStr(const string& str){  int pos = strHash(str),cnt = 0;  while(!store[pos].empty()){  pos++;  cnt++;  if(cnt > N) return;  pos %= N;  }  store[pos] = str;  }  int findStr(const string& str){  int pos = strHash(str),cnt = 0;  while(store[pos] != str){  pos++;  cnt++;  if(cnt > N) return -1;  pos %= N;  }  return pos;  }  (2)  void insertSort(int arr[],int len);  void shellSort(int arr[],int len);  void bubbleSort(int arr[],int len);  void quickSort(int arr[],int len);  void quickDfs(int l,int r);  void selectSort(int arr[],int len);  void heapSort(int arr[],int len);  void heapInsert(int val);  int heapPop();  void mergeSort(int arr[],int len);  void mergeDfs(int l,int r);  void radixSort(int arr[],int len);  int qpow(int x,int y);  3. 测试用例设计  (1)  string name[10] = {"XiaoMing","ZhangSan","LiSi","ErGou","WangWu",  "Yanzi","LuYiSi","LiuJiangJun","SiShen","XiaoChou"};  (2)  int mock[MOCK\_N] = {45,33,59,28,128,256,9,3,234,1024};  三、**实验软硬件环境**  {  "version": "2.0.0",  "tasks": [  {  "type": "shell",  "label": "C/C++: g++.exe 生成活动文件",  "command": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe",  "args": [  "-g",  "${file}",  "-o",  "${fileDirname}\\${fileBasenameNoExtension}.exe"  ],  "options": {  "cwd": "E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin"  },  "problemMatcher": [  "$gcc"  ],  "group": {  "kind": "build",  "isDefault": true  },  "detail": "编译器: E:\\Code\\C++\\x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0\\mingw64\\bin\\g++.exe"  }  ]  }  第二部分：实验过程与结果  **一、实验说明（所建工程文件等的说明）**    **二、调试说明（调试手段、过程及结果分析）**  vscode 断点调试 codeRunner 运行  过程顺利  结果无误   1. **软件测试（测试效果.界面、综合分析和结论）**   (1)    (2)      第三部分：实验小结、收获与体会  **对于Hans和内部排序的操作有了更深入的认识，增强了代码能力，提升了算法思维,对于调试程序的操作更为熟悉，学习了出现问题如何分析原因追溯源头，培养了不骄不躁，埋下头学习的刻苦优秀品质。**  第四部分：附源程序  (1)  #include <bits/stdc++.h>  using namespace std;  const int MOD = 23;  const int N = 128;  string name[10] = {"XiaoMing","ZhangSan","LiSi","ErGou","WangWu",  "Yanzi","LuYiSi","LiuJiangJun","SiShen","XiaoChou"};  string store[N];  int strHash(const string& str){  int res = 1;  for(auto ch: str){  res = (res + (ch - 'A')) % MOD;  }  return res;  }  void pushStr(const string& str){  int pos = strHash(str),cnt = 0;  while(!store[pos].empty()){  pos++;  cnt++;  if(cnt > N) return;  pos %= N;  }  store[pos] = str;  }  int findStr(const string& str){  int pos = strHash(str),cnt = 0;  while(store[pos] != str){  pos++;  cnt++;  if(cnt > N) return -1;  pos %= N;  }  return pos;  }  int main(){  for(auto s: name) pushStr(s);  for(auto s: name){  cout << strHash(s) << " " << findStr(s) << endl;  }  return 0;  }  (2)  #include <bits/stdc++.h>  using namespace std;  #define lson(k) (k << 1)  #define rson(k) (k << 1 | 1)  const int N = 1024;  const int MOCK\_N = 10;  int mock[MOCK\_N] = {45,33,59,28,128,256,9,3,234,1024};  int quickArr[N],quickN;  int heapTree[N],heapN;  bool heapUsed[N];  int mergeArr[N],mergeN;  inline void printArr(int arr[],int len){  for(int i = 0;i < len;i++) printf("%d ",arr[i]);  printf("\n");  }  void insertSort(int arr[],int len);  void shellSort(int arr[],int len);  void bubbleSort(int arr[],int len);  void quickSort(int arr[],int len);  void quickDfs(int l,int r);  void selectSort(int arr[],int len);  void heapSort(int arr[],int len);  void heapInsert(int val);  int heapPop();  void mergeSort(int arr[],int len);  void mergeDfs(int l,int r);  void radixSort(int arr[],int len);  int qpow(int x,int y);  void show(int arr[],int len);  int main(){  show(mock,MOCK\_N);  return 0;  }  void show(int arr[],int len){  printf("insertSort:\n");  insertSort(arr,len);  printf("shellSort:\n");  shellSort(arr,len);  printf("bubbleSort:\n");  bubbleSort(arr,len);  printf("quickSort:\n");  quickSort(arr,len);  printf("selectSort:\n");  selectSort(arr,len);  printf("heapSort:\n");  heapSort(arr,len);  printf("mergeSort:\n");  mergeSort(arr,len);  printf("radixSort:\n");  radixSort(arr,len);  }  void insertSort(int arr[],int len){  int n = 0;  int list[N];  for(int i = 0;i < len;i++){  int pos = n;  while(pos && list[pos - 1] > arr[i]){  list[pos] = list[pos - 1];  pos--;  }  list[pos] = arr[i];  n++;  printArr(list,n);  }  }  void shellSort(int arr[],int len){  int step = len >> 1;  while(step){  for(int i = step;i < len;i++){  int pos = i;  while(pos - step >= 0 && arr[pos - step] > arr[pos]){  swap(arr[pos - step],arr[pos]);  pos -= step;  }  }  step >>= 1;  printArr(arr,len);  }  }  void bubbleSort(int arr[],int len){  for(int i = len;i > 0;i--){  for(int j = 0;j < i;j++)  if(arr[j] < arr[j - 1]) swap(arr[j],arr[j - 1]);    printArr(arr,len);  }  }  void quickSort(int arr[],int len){  quickN = len;  for(int i = 0;i < len;i++) quickArr[i] = arr[i];  quickDfs(0,len - 1);  }  void quickDfs(int l,int r){  if(l >= r) return;  int divider = quickArr[l];  int s = l,t = r;  while(s < t){  while(s < t && quickArr[t] > divider) t--;  quickArr[s] = quickArr[t];  while(s < t && quickArr[s] < divider) s++;  quickArr[t] = quickArr[s];  }  quickArr[s] = divider;  printArr(quickArr,quickN);  quickDfs(l,s);  quickDfs(s + 1,r);  }  void selectSort(int arr[],int len){  for(int i = 0;i < len;i++){  int pos = i;  for(int j = i + 1;j < len;j++)  if(arr[j] < arr[pos]) pos = j;  swap(arr[i],arr[pos]);  printArr(arr,len);  }  }  void heapSort(int arr[],int len){  memset(heapUsed,0,sizeof(heapUsed));  for(int i = 0;i < len;i++) heapInsert(arr[i]);  for(int i = 0;i < len;i++) printf("%d ",heapPop());  printf("\n");  }  void heapInsert(int val){  heapTree[++heapN] = val;  heapUsed[heapN] = true;  int cur = heapN;  int fa = cur >> 1;  while(fa && heapTree[fa] > heapTree[cur]){  swap(heapTree[fa],heapTree[cur]);  cur = fa;  fa = cur >> 1;  }  printArr(heapTree,heapN);  }  int heapPop(){  int res = heapTree[1];  int cur = 1;  while(heapUsed[lson(cur)] || heapUsed[rson(cur)]){  heapUsed[cur] = true;  if(heapUsed[lson(cur)] && heapUsed[rson(cur)]){  if(heapTree[lson(cur)] > heapTree[rson(cur)]){  heapTree[cur] = heapTree[rson(cur)];  cur = rson(cur);  } else {  heapTree[cur] = heapTree[lson(cur)];  cur = lson(cur);  }  } else if(heapUsed[lson(cur)]) {  heapTree[cur] = heapTree[lson(cur)];  cur = lson(cur);  } else {  heapTree[cur] = heapTree[rson(cur)];  cur = rson(cur);  }  heapUsed[cur] = false;  }  return res;  }  void mergeSort(int arr[],int len){  mergeN = len;  for(int i = 0;i < len;i++) mergeArr[i] = arr[i];  mergeDfs(0,mergeN - 1);  }  void mergeDfs(int l,int r){  if(l >= r) return;  int mid = (l + r) >> 1;  mergeDfs(l,mid);  mergeDfs(mid + 1,r);  queue<int> q1,q2;  for(int i = l;i <= r;i++){  if(i <= mid) q1.push(mergeArr[i]);  else q2.push(mergeArr[i]);  }  for(int i = l;i <= r;i++){  if(q2.empty() || (!q1.empty() && q1.front() < q2.front())){  mergeArr[i] = q1.front();  q1.pop();  } else {  mergeArr[i] = q2.front();  q2.pop();  }  }  printArr(mergeArr,mergeN);  }  void radixSort(int arr[],int len){  queue <int> q[10];  int bit = 0;  for(int i = 0;i < len;i++){  int tmp = 1,cur = arr[i];  while(cur / 10){  tmp++;  cur /= 10;  }  bit = max(bit,tmp);  }  for(int i = 0;i < bit;i++){  for(int j = 0;j < len;j++){  int tt = (arr[j] % qpow(10,i + 1)) / qpow(10,i);  q[(arr[j] % qpow(10,i + 1)) / qpow(10,i)].push(arr[j]);  }  int pos = 0;  for(int j = 0;j < 10;j++){  while(!q[j].empty()){  arr[pos++] = q[j].front();  q[j].pop();  }  }  printArr(arr,len);  }  }  int qpow(int x,int y){  int res = 1;  while(y){  if(y & 1) res = res \* x;  x = x \* x;  y >>= 1;  }  return res;  } |