EX 01.A

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

typedef struct Node {
int data;
struct Node *left, *right;
} Node;

Node* newNode(int x) {
Node *n = (Node*)malloc(sizeof(Node));
n->data = x;
n->left = n->right = NULL;
return n;
}

void inorder(Node *root) {
if (!root) return;
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}

void preorder(Node *root) {
if (!root) return;
printf("%d ", root->data);
preorder(root->left);
preorder(root->right);
}

void postorder(Node *root) {
if (!root) return;
postorder(root->left);
postorder(root->right);
printf("%d ", root->data);
}

int main() {
Node *root = newNode(1);
root->left = newNode(2);
root->right = newNode(3);
root->left->left = newNode(4);
root->left->right = newNode(5);

printf("Inorder: ");
inorder(root);
printf("\nPreorder: ");
preorder(root);
```

```
printf("\nPostorder: ");
postorder(root);
return 0;
}
```

EX 1.b

```
#include <stdio.h>

int fib(int n) {
if (n <= 1) return n;
return fib(n-1) + fib(n-2);
}

int main() {
int n;
scanf("%d", &n);
for (int i = 0; i < n; i++)
printf("%d ", fib(i));
}
```

Ex.2.a

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

typedef struct Node {
int data;
struct Node *left, *right;
} Node;

typedef struct {
Node *a[MAX];
int top;
} Stack;

Node* newNode(int x) {
Node *n = (Node*)malloc(sizeof(Node));
n->data = x;
n->left = n->right = NULL;
return n;
}

void push(Stack *s, Node *n) { s->a[++s->top] = n; }
```

```
Node* pop(Stack *s) { return s->a[s->top--]; }
int empty(Stack *s) { return s->top < 0; }

void inorderIter(Node *root) {
Stack s; s.top = -1;
Node *cur = root;
while (cur || !empty(&s)) {
while (cur) {
push(&s, cur);
cur = cur->left;
}
cur = pop(&s);
printf("%d ", cur->data);
cur = cur->right;
}
}

void preorderIter(Node *root) {
if (!root) return;
Stack s; s.top = -1;
push(&s, root);
while (!empty(&s)) {
Node *n = pop(&s);
printf("%d ", n->data);
if (n->right) push(&s, n->right);
if (n->left)  push(&s, n->left);
}
}

void postorderIter(Node *root) {
if (!root) return;
Stack s1, s2;
s1.top = s2.top = -1;
push(&s1, root);
while (!empty(&s1)) {
Node *n = pop(&s1);
push(&s2, n);
if (n->left)  push(&s1, n->left);
if (n->right) push(&s1, n->right);
}
while (!empty(&s2))
printf("%d ", pop(&s2)->data);
}

int main() {
Node *root = newNode(1);
root->left = newNode(2);
root->right = newNode(3);
```

```
root->left->left = newNode(4);
root->left->right = newNode(5);

printf("Inorder: ");
inorderIter(root);
printf("\nPreorder: ");
preorderIter(root);
printf("\nPostorder: ");
postorderIter(root);
}
```

EX2.B

```
#include <stdio.h>
int main() {
int n, a = 0, b = 1, c;
scanf("%d", &n);
for (int i = 0; i < n; i++) {
printf("%d ", a);
c = a + b;
a = b;
b = c;
}
}
```

Ex.3A

```
#include <stdio.h>

void sw(int *x,int *y){
int t=*x;*x=*y;*y=t;
}

int part(int a[],int l,int h){
int p=a[h],i=l-1,j;
for(j=l;j<h;j++)
if(a[j]<p){ i++; sw(&a[i],&a[j]); }
sw(&a[i+1],&a[h]);
return i+1;
}

void qs(int a[],int l,int h){
if(l<h){
int p=part(a,l,h);
qs(a,l,p-1);
qs(a,p+1,h);
}
```

```
}

int main(){
int n,i;
scanf("%d",&n);
int a[n];
for(i=0;i<n;i++) scanf("%d",&a[i]);
qs(a,0,n-1);
for(i=0;i<n;i++) printf("%d ",a[i]);
}
```

Ex3.B

```
#include <stdio.h>

void sw(int *a,int *b){
int t=*a;*a=*b;*b=t;
}

int part(int a[],int l,int h){
int p=a[h],i=l-1,j;
for(j=l;j<h;j++)
if(a[j]<p){i++;sw(&a[i],&a[j]);}
sw(&a[i+1],&a[h]);
return i+1;
}

void qs(int a[],int l,int h){
if(l<h){
int p=part(a,l,h);
qs(a,l,p-1);
qs(a,p+1,h);
}
}

int main(){
int n,i;
scanf("%d",&n);
int a[n];
for(i=0;i<n;i++) scanf("%d",&a[i]);
qs(a,0,n-1);
for(i=0;i<n;i++) printf("%d ",a[i]);
}
```

Ex.4

```
#include <stdio.h>
```

```
#include <stdlib.h>

typedef struct Node {
int d;
struct Node *l,*r;
} Node;

Node* new(int x){
Node *n = (Node*)malloc(sizeof(Node));
n->d = x;
n->l = n->r = NULL;
return n;
}

Node* ins(Node *root,int x){
if(!root) return new(x);
if(x < root->d) root->l = ins(root->l,x);
else if(x > root->d) root->r = ins(root->r,x);
return root;
}

void in(Node *root){
if(!root) return;
in(root->l);
printf("%d ",root->d);
in(root->r);
}

void pre(Node *root){
if(!root) return;
printf("%d ",root->d);
pre(root->l);
pre(root->r);
}

void post(Node *root){
if(!root) return;
post(root->l);
post(root->r);
printf("%d ",root->d);
}

Node* sea(Node *root,int k){
if(!root || root->d==k) return root;
if(k < root->d) return sea(root->l,k);
return sea(root->r,k);
}
```

```
int main(){
int n,i,x,k;
Node *root = NULL;
scanf("%d",&n);
for(i=0;i<n;i++){
scanf("%d",&x);
root = ins(root,x);
}
in(root);   printf("\n");
pre(root);  printf("\n");
post(root); printf("\n");
scanf("%d",&k);
if(sea(root,k)) printf("found");
else printf("not found");
}
```

Ex.5

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

#define R 0
#define B 1

typedef struct Node {
int d,c;
struct Node *l,*r,*p;
} Node;

Node* newN(int x){
Node *n = (Node*)malloc(sizeof(Node));
n->d=x; n->c=R;
n->l=n->r=n->p=NULL;
return n;
}

void rotL(Node **root,Node *x){
Node *y=x->r;
x->r=y->l;
if(y->l) y->l->p=x;
y->p=x->p;
if(!x->p) *root=y;
else if(x==x->p->l) x->p->l=y;
else x->p->r=y;
y->l=x;
x->p=y;
}
```

```
void rotR(Node **root,Node *y){
Node *x=y->l;
y->l=x->r;
if(x->r) x->r->p=y;
x->p=y->p;
if(!y->p) *root=x;
else if(y==y->p->l) y->p->l=x;
else y->p->r=x;
x->r=y;
y->p=x;
}

void fix(Node **root,Node *z){
while(z->p && z->p->c==R){
Node *g=z->p->p;
if(z->p==g->l){
Node *u=g->r;
if(u && u->c==R){
z->p->c=u->c=B;
g->c=R;
z=g;
}else{
if(z==z->p->r){
z=z->p;
rotL(root,z);
}
z->p->c=B;
g->c=R;
rotR(root,g);
}
}else{
Node *u=g->l;
if(u && u->c==R){
z->p->c=u->c=B;
g->c=R;
z=g;
}else{
if(z==z->p->l){
z=z->p;
rotR(root,z);
}
z->p->c=B;
g->c=R;
rotL(root,g);
}
}
}
```

```
(*root)->c=B;
}

void ins(Node **root,int x){
Node *z=newN(x),*y=NULL,*t=*root;
while(t){
y=t;
if(z->d<t->d) t=t->l;
else t=t->r;
}
z->p=y;
if(!y) *root=z;
else if(z->d<y->d) y->l=z;
else y->r=z;
fix(root,z);
}

void in(Node *t){
if(!t) return;
in(t->l);
printf("%d(%c) ",t->d,t->c==R?'R':'B');
in(t->r);
}

int main(){
int n,x,i;
Node *root=NULL;
scanf("%d",&n);
for(i=0;i<n;i++){
scanf("%d",&x);
ins(&root,x);
}
in(root);
}
```

EX.6

```
#include <stdio.h>

void sw(int*a,int*b){int t=*a;*a=*b;*b=t;}

void heapify(int a[],int n,int i){
int largest=i,l=2*i+1,r=2*i+2;
if(l<n && a[l]>a[largest]) largest=l;
if(r<n && a[r]>a[largest]) largest=r;
if(largest!=i){ sw(&a[i],&a[largest]); heapify(a,n,largest); }
}
```

```
void build(int a[],int n){ for(int i=n/2-1;i>=0;i--) heapify(a,n,i); }

void heapsort(int a[],int n){ build(a,n); for(int i=n-1;i>0;i--){ sw(&a[0],&a[i]); heapify(a,i,0); } }

int main(){
int n; if(scanf("%d",&n)!=1) return 0;
int a[n];
for(int i=0;i<n;i++) scanf("%d",&a[i]);

printf("Original: ");
for(int i=0;i<n;i++) printf("%d ",a[i]);
printf("\n");

build(a,n);
printf("Max Heap: ");
for(int i=0;i<n;i++) printf("%d ",a[i]);
printf("\n");

heapsort(a,n);
printf("Sorted: ");
for(int i=0;i<n;i++) printf("%d ",a[i]);
printf("\n");
}
```

Ex.7

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

typedef struct F {
int k,deg,mark;
struct F *p,*c,*l,*r;
} F;

typedef struct {
F *min;
int n;
} H;

F* newN(int x){
F *t = (F*)malloc(sizeof(F));
t->k=x; t->deg=0; t->p=t->c=NULL;
t->l=t->r=t;
t->mark=0;
return t;
}
```

```
H* newH(){
H *h = (H*)malloc(sizeof(H));
h->min=NULL; h->n=0;
return h;
}

void insert(H *h,F *x){
if(!h->min) h->min=x;
else{
x->l=h->min;
x->r=h->min->r;
h->min->r->l=x;
h->min->r=x;
if(x->k < h->min->k) h->min=x;
}
h->n++;
}

void link(H *h,F *y,F *x){
y->l->r=y->r;
y->r->l=y->l;
y->p=x;
if(!x->c) x->c=y,y->l=y->r=y;
else{
y->l=x->c;
y->r=x->c->r;
x->c->r->l=y;
x->c->r=y;
}
x->deg++;
y->mark=0;
}

void cons(H *h){
int i,D=50;
F *A[50]={0},*w=h->min,*start,*x,*y,*tmp;
if(!w) return;
start=w;
do{
x=w;
int d=x->deg;
while(A[d]){
y=A[d];
if(x->k > y->k){ tmp=x; x=y; y=tmp; }
link(h,y,x);
A[d]=NULL;
d++;
}
```

```
A[d]=x;
w=w->r;
}while(w!=start);
h->min=NULL;
for(i=0;i<D;i++)
if(A[i]){
if(!h->min){
h->min=A[i];
h->min->l=h->min->r=h->min;
}else{
A[i]->l=h->min;
A[i]->r=h->min->r;
h->min->r->l=A[i];
h->min->r=A[i];
if(A[i]->k < h->min->k) h->min=A[i];
}
}
}

F* extractMin(H *h){
F *z=h->min;
if(z){
if(z->c){
F *c=z->c,*next,*start=c;
do{
next=c->r;
c->l=h->min;
c->r=h->min->r;
h->min->r->l=c;
h->min->r=c;
c->p=NULL;
c=next;
}while(c!=start);
}
z->l->r=z->r;
z->r->l=z->l;
if(z==z->r) h->min=NULL;
else{
h->min=z->r;
cons(h);
}
h->n--;
}
return z;
}

int main(){
H *h = newH();
```

```
insert(h,newN(10));
insert(h,newN(3));
insert(h,newN(15));
insert(h,newN(6));
printf("Min: %d\n",h->min->k);
F *m = extractMin(h);
printf("Extracted Min: %d\n",m->k);
printf("New Min: %d\n",h->min->k);
}
```

Ex.8

```
#include <stdio.h>
#define MAX 20

int a[MAX][MAX],vis[MAX],q[MAX];
int n,f=0,r=-1;

void en(int v){ q[++r]=v; }
int de(){ return (f<=r)? q[f++] : -1; }

void bfs(int s){
int v,i;
for(i=0;i<n;i++) vis[i]=0;
vis[s]=1; en(s);
printf("BFS: ");
while((v=de())!=-1){
printf("%d ",v);
for(i=0;i<n;i++)
if(a[v][i] && !vis[i]){
vis[i]=1;
en(i);
}
}
}

void dfs(int v){
int i;
printf("%d ",v);
vis[v]=1;
for(i=0;i<n;i++)
if(a[v][i] && !vis[i])
dfs(i);
}

int main(){
int i,j,s;
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
scanf("%d",&s);
bfs(s);
printf("\nDFS: ");
for(i=0;i<n;i++) vis[i]=0;
dfs(s);
}
```

Ex.9.A Prim’s Algorithm (Adjacency Matrix)

```
#include <stdio.h>
#define INF 999
#define MAX 20

int g[MAX][MAX],vis[MAX],n;

void prim(){
int i,j,ec=0,mc=0;
vis[0]=1;
while(ec<n-1){
int mn=INF,u=-1,v=-1;
for(i=0;i<n;i++)
if(vis[i])
for(j=0;j<n;j++)
if(!vis[j] && g[i][j]<mn){
mn=g[i][j]; u=i; v=j;
}
printf("%d -> %d cost = %d\n",u,v,mn);
vis[v]=1;
mc+=mn;
ec++;
}
printf("Min cost = %d\n",mc);
}

int main(){
int i,j;
scanf("%d",&n);
for(i=0;i<n;i++)
for(j=0;j<n;j++){
scanf("%d",&g[i][j]);
if(g[i][j]==0) g[i][j]=INF;
}
prim();
}
```

B.Kruskal’s Algorithm

```
#include <stdio.h>
#define MAX 30
#define INF 999

int p[MAX], c[MAX][MAX], n;

int find(int i){
while(p[i]) i=p[i];
return i;
}

int uni(int i,int j){
if(i!=j){ p[j]=i; return 1; }
return 0;
}

int main(){
int i,j,a,b,u,v,edges=1,min,mc=0;
scanf("%d",&n);
for(i=1;i<=n;i++)
for(j=1;j<=n;j++){
scanf("%d",&c[i][j]);
if(c[i][j]==0) c[i][j]=INF;
}
while(edges<n){
min=INF;
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(c[i][j]<min){
min=c[i][j];
a=u=i; b=v=j;
}
u=find(u); v=find(v);
if(uni(u,v)){
printf("%d -> %d cost = %d\n",a,b,min);
mc+=min;
edges++;
}
c[a][b]=c[b][a]=INF;
}
printf("Min cost = %d\n",mc);
}
```

10.A

```
#include <stdio.h>
#include <limits.h>

#define V 5

int minD(int dist[], int vis[]){
int min = INT_MAX, idx = -1, i;
for(i = 0; i < V; i++)
if(!vis[i] && dist[i] <= min){
min = dist[i];
idx = i;
}
return idx;
}

void dijkstra(int g[V][V], int src){
int dist[V], vis[V] = {0}, i, j;
for(i = 0; i < V; i++) dist[i] = INT_MAX;
dist[src] = 0;

for(i = 0; i < V-1; i++){
int u = minD(dist, vis);
vis[u] = 1;
for(j = 0; j < V; j++)
if(!vis[j] && g[u][j] && dist[u] != INT_MAX &&
dist[u] + g[u][j] < dist[j])
dist[j] = dist[u] + g[u][j];
}

printf("Vertex\tDistance from Source\n");
for(i = 0; i < V; i++)
printf("%d\t\t%d\n", i, dist[i]);
}

int main(){
int g[V][V] = {
{0, 10, 0, 30, 100},
{10, 0, 50, 0, 0},
{0, 50, 0, 20, 10},
{30, 0, 20, 0, 60},
{100,0,10, 60, 0}
};
dijkstra(g, 0);
}
```

Ex.10.b

```
#include <stdio.h>
#include <limits.h>

#define V 5
#define E 8

struct Edge { int u,v,w; } e[E] = {
{0,1,-1},
{0,2, 4},
{1,2, 3},
{1,3, 2},
{1,4, 2},
{3,2, 5},
{3,1, 1},
{4,3,-3}
};

void bf(int s){
int d[V],i,j;
for(i=0;i<V;i++) d[i]=INT_MAX;
d[s]=0;

for(i=0;i<V-1;i++)
for(j=0;j<E;j++)
if(d[e[j].u]!=INT_MAX && d[e[j].u]+e[j].w<d[e[j].v])
d[e[j].v]=d[e[j].u]+e[j].w;

for(j=0;j<E;j++)
if(d[e[j].u]!=INT_MAX && d[e[j].u]+e[j].w<d[e[j].v]){
printf("Negative cycle\n");
return;
}

printf("Vertex\tDistance from Source\n");
for(i=0;i<V;i++)
printf("%d\t\t%d\n",i,d[i]);
}

int main(){
bf(0);
}
```

Ex.11

```
#include <stdio.h>
#include <limits.h>
```

```
#define MAX 100

int mco(int p[],int n){
int m[MAX][MAX],i,j,L,k,q;
for(i=1;i<n;i++) m[i][i]=0;
for(L=2;L<n;L++)
for(i=1;i<n-L+1;i++){
j=i+L-1;
m[i][j]=INT_MAX;
for(k=i;k<j;k++){
q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
if(q<m[i][j]) m[i][j]=q;
}
}
return m[1][n-1];
}

int main(){
int n,i,p[MAX];
scanf("%d",&n);
for(i=0;i<=n;i++) scanf("%d",&p[i]);
printf("%d",mco(p,n+1));
}
```

Ex.12.A

```
#include <stdio.h>

typedef struct{
int s,f,id;
} Act;

void sort(Act a[],int n){
int i,j; Act t;
for(i=0;i<n-1;i++)
for(j=0;j<n-1-i;j++)
if(a[j].f>a[j+1].f){
t=a[j]; a[j]=a[j+1]; a[j+1]=t;
}
}

void select(Act a[],int n){
int i=0,j;
printf("A%d (%d,%d)\n",a[0].id,a[0].s,a[0].f);
for(j=1;j<n;j++)
if(a[j].s>=a[i].f){
```

```
printf("A%d (%d,%d)\n",a[j].id,a[j].s,a[j].f);
i=j;
}
}

int main(){
int n,i;
scanf("%d",&n);
Act a[n];
for(i=0;i<n;i++){
scanf("%d%d",&a[i].s,&a[i].f);
a[i].id=i+1;
}
sort(a,n);
select(a,n);
}
```

EX.12.B

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

typedef struct N { char c; int f; struct N *l,*r; } N;
typedef struct { int n; N *a[100]; } H;

N* nnode(char c,int f){ N *t=malloc(sizeof(N)); t->c=c; t->f=f; t->l=t->r=NULL; return t; }
void sw(N**x,N**y){ N*t=*x;*x=*y;*y=t; }

void heapify(H*h,int i){
int s=i,L=2*i+1,R=2*i+2;
if(L<h->n && h->a[L]->f < h->a[s]->f) s=L;
if(R<h->n && h->a[R]->f < h->a[s]->f) s=R;
if(s!=i){ sw(&h->a[s],&h->a[i]); heapify(h,s); }
}

N* extract(H*h){
N*r=h->a[0]; h->a[0]=h->a[--h->n]; heapify(h,0); return r;
}

void insert(H*h,N*x){
int i=h->n++;
while(i && x->f < h->a[(i-1)/2]->f){ h->a[i]=h->a[(i-1)/2]; i=(i-1)/2; }
h->a[i]=x;
}

N* build(char d[],int f[],int n){
H h; h.n=n; for(int i=0;i<n;i++) h.a[i]=nnode(d[i],f[i]);
for(int i=n/2-1;i>=0;i--) heapify(&h,i);
```

```
while(h.n>1){
N*l=extract(&h), *r=extract(&h);
N* t=nnode('$', l->f + r->f); t->l=l; t->r=r; insert(&h,t);
}
return extract(&h);
}

void printCodes(N*r,int code[],int top){
if(!r->l && !r->r){ printf("%c: ", r->c); for(int i=0;i<top;i++) printf("%d",code[i]); printf("\n"); return; }
code[top]=0; if(r->l) printCodes(r->l, code, top+1);
code[top]=1; if(r->r) printCodes(r->r, code, top+1);
}

int main(){
int n; if(scanf("%d",&n)!=1) return 0;
char d[n]; int f[n], code[100];
for(int i=0;i<n;i++){ scanf(" %c%d",&d[i],&f[i]); }
N *root = build(d,f,n);
printCodes(root, code, 0);
}
```