

# SOLUTION

November 5, 2009

Name \_\_\_\_\_

## ECE264 Advanced C Programming Exam 2

Solve the following problems. The number of points for each problem is shown next to the problem and in the table below. The outcomes corresponding to each problem are also shown. Use only the space provided to solve each problem.

Problem	Points	Outcome
1 (a)	/ 5	3
(b)	/ 5	
(c)	/ 5	
(d)	/ 5	
(e)	/ 5	
(f)	/ 5	
(g)	/ 5	
(h)	/ 5	
(i)	/ 5	
(j)	/ 5	
2 (a)	/ 10	
(b)	/ 10	
3 (a)	/ 15	
(b)	/ 15	
Total	/ 100	

**Problem 1 (50 points)**

In the following program the calls to the function `printlist()` are marked (a), (b), ..., (j). Specify what the program will print in each one of these calls.

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

#define N1 3

struct list
{
    int data;
    struct list *next;
};

int main(void)
{
    int i;
    struct list *head1,*head2;
    struct list *listfunc1(struct list *, int), listfunc2(struct list *, int);
    struct list *listfunc3(struct list *, struct list *), listfunc4(struct list *);
    void printlist(int, struct list *);

    head1=NULL;
    for(i=1;i<=N1;i++) head1=listfunc1(head1,i);
    printlist(1,head1); /* (a) */

    head2=NULL;
    for(i=1;i<=N1;i++) head2=listfunc2(head2,i);
    printlist(2,head2); /* (b) */

    head1=listfunc3(head1,head2);
    printlist(1,head1); /* (c) */
    printlist(2,head2); /* (d) */

    head2=listfunc4(head2);
    printlist(1,head1); /* (e) */
    printlist(2,head2); /* (f) */

    head2=listfunc4(head2);
    printlist(1,head1); /* (g) */
    printlist(2,head2); /* (h) */

    head2=listfunc4(head2);
    printlist(1,head1); /* (i) */
    printlist(2,head2); /* (j) */
}
```

```
void printlist(int ind, struct list *head)
{
    struct list *p;

    printf("\nlist%d:",ind);
    for(p=head;p!=NULL;p=p->next)
        printf(" %d",p->data);
    printf("\n");
}

struct list *listfunc1(struct list *head, int data)
{
    struct list *p;

    p=malloc(sizeof(struct list));
    assert(p!=NULL);
    p->data=data;

    p->next=head;
    head=p;

    return(head);
}

struct list *listfunc2(struct list *head, int data)
{
    struct list *p,*q;

    p=malloc(sizeof(struct list));
    assert(p!=NULL);
    p->data=data;
    p->next=NULL;

    if(head==NULL) head=p;
    else
    {
        for(q=head;q->next!=NULL;q=q->next);
        q->next=p;
    };

    return(head);
}
```

```
struct list *listfunc3(struct list *head1, struct list *head2)
{
    struct list *p,*q;

    if(head1==NULL) head1=head2;
    else
    {
        for(q=head1;q->next!=NULL;q=q->next);
        q->next=head2;
    };

    return(head1);
}
```

```
struct list *listfunc4(struct list *head)
{
    struct list *p,*q;

    if(head!=NULL)
    {
        for(p=head;p->next!=NULL;p=p->next)
            q=p;

        if(p!=head)
        {
            free(p);
            q->next=NULL;
        };
    };

    return(head);
}
```

- (a) In call (a) to printlist() the program will print:  
list1: 3 2 1
- (b) In call (b) to printlist() the program will print:  
list2: 1 2 3
- (c) In call (c) to printlist() the program will print:  
list1: 3 2 1 1 2 3
- (d) In call (d) to printlist() the program will print:  
list2: 1 2 3
- (e) In call (e) to printlist() the program will print:  
list1: 3 2 1 1 2
- (f) In call (f) to printlist() the program will print:  
list2: 1 2
- (g) In call (g) to printlist() the program will print:  
list1: 3 2 1 1
- (h) In call (h) to printlist() the program will print:  
list2: 1
- (i) In call (i) to printlist() the program will print:  
list1: 3 2 1 1
- (j) In call (j) to printlist() the program will print:  
list2: 1

**Problem 2 (20 points)**

In the following program the calls to the function printqu() are marked (a) and (b). Specify what the program will print in each one of these calls.

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

#define N1 5
#define N2 3

struct elem
{
    int data;
    struct elem *next;
};

struct queue
{
    struct elem *front,*rear;
};

int main(void)
{
    int i,j;
    struct queue qu;
    void insert(struct queue *, int),printqu(struct queue);
    int delete(struct queue *);

    qu.front=NULL;
    qu.rear=NULL;

    for(i=0;i<N1;i++) insert(&qu,i);
    printqu(qu); /* (a) */
    for(i=0;i<N2;i++)
    {
        j=delete(&qu);
        insert(&qu,j);
    };
    printqu(qu); /* (b) */
    printf("\n");
}
```

```
void printqu(struct queue qu)
{
    struct elem *p;

    printf("\nqueue:");
    for(p=qu.front;p!=NULL;p=p->next) printf(" %d",p->data);
}

void insert(struct queue *pqu, int n)
{
    struct elem *p;

    p=malloc(sizeof(struct elem));
    assert(p!=NULL);

    p->data=n;
    p->next=NULL;

    if(pqu->front==NULL) pqu->front=p;
    else pqu->rear->next=p;

    pqu->rear=p;
}

int delete(struct queue *pqu)
{
    int n;
    struct elem *fr;

    assert(pqu->front!=NULL);
    n=pqu->front->data;
    fr=pqu->front;
    pqu->front=pqu->front->next;
    if(pqu->front==NULL) pqu->rear=NULL;
    free(fr);
    return(n);
}
```

(a) In call (a) to printqu() the program will print:

queue: 0 1 2 3 4

(b) In call (b) to printqu() the program will print:

queue: 3 4 0 1 2

### Problem 3 (30 points)

Show the tree that the following program will create and specify what the program will print for the input in part (a), and for the input in part (b).

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

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

int main(void)
{
    int n,i,data;
    struct node *root;
    void someorder(struct node *);
    struct node *addnewnode(struct node *, int);

    root=NULL;

    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&data);
        root=addnewnode(root,data);
    };
    printf("\ntree elements:");
    someorder(root);
    printf("\n");
}

void someorder(struct node *root)
{
    if(root!=NULL)
    {
        someorder(root->right);
        printf("\n%d",root->data);
        someorder(root->left);
    }
}
```

```
struct node *addnewnode(struct node * root, int data)
{
    struct node *p;
    struct node *allocnewnode(int);

    if(root==NULL)
    {
        root=allocnewnode(data);
        return(root);
    };

    for(p=root;1;)
    {
        if(data<=p->data)
        {
            if(p->left==NULL)
            {
                p->left=allocnewnode(data);
                return(root);
            };
            p=p->left;
        }
        else
        {
            if(p->right==NULL)
            {
                p->right=allocnewnode(data);
                return(root);
            };
            p=p->right;
        };
    };
}

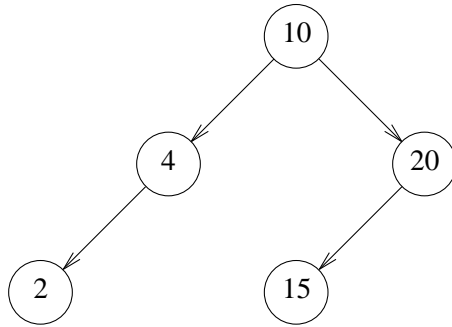
struct node *allocnewnode(int data)
{
    struct node *pnew;

    pnew=malloc(sizeof(struct node));
    assert(pnew!=NULL);
    pnew->data=data;
    pnew->left=NULL;
    pnew->right=NULL;
    return(pnew);
}
```

(a)

5  
10 4 20 15 2

The tree:



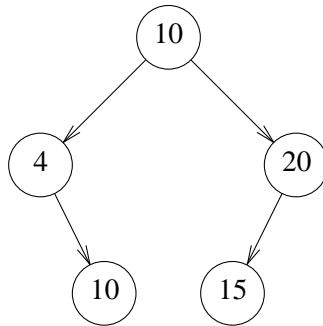
The program will print:

tree elements:  
20  
15  
10  
4  
2

(b)

5  
10 4 20 15 10

The tree:



The program will print:

tree elements:  
20  
15  
10  
10  
4