ABHISHEK SHARMA

CS 2ND YEAR

SECTION : “I”

ROLL NO.: 01

ENROLLMENT NO.: 12019009001127

DATA STRUCTURE AND ALGORITHM LABORATORY

WEEK : 7

ASSIGNMENT : 7

DATE : 25.08.2020

HACKERRANK ID : 12019009001127_I

Q1. Insert an element in a specific position in doubly linked list

Code :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *link1;
    struct node *link2;
};
int main()
{
    int i,d,pos,l;
    char a[50];
    scanf("%d",&d);
    scanf("%d",&pos);
    getchar();
    fgets(a,50,stdin);
    l=(strlen(a)/2);
    int ar[l];
    int dd=0,r;
    for(i=0;i<strlen(a);i++)
    {
        if((int)a[i]>=48)
        {
            r=(int)a[i]-48;
            ar[dd++]=r;
        }
    }
    l++;
    struct node *p,*h;
    p=h=(struct node*)malloc(sizeof(struct node));
    p->link1=NULL;
    p->data=ar[0];
    for(i=1;i<l;i++)
    {
        p->link2=(struct node*)malloc(sizeof(struct node));
        p=p->link2;
        p->data=ar[i];
    }
    p->link2=NULL;
    p=h;
    int c=1;
    struct node *x;
    x=(struct node*)malloc(sizeof(struct node));
    if(pos==1)
    {
        x->data=d;
        h->link1=x;
```

```
        x->link2=h;
        x->link1=NULL;
        h=x;
    }
    else if(pos>l)
    {
        while(p->link1!=NULL)
        {
            p=p->link1;
        }
        x->data=d;
        p->link2=x;
        x->link1=p;
        x->link2=NULL;
    }
    else
    {
        struct node *q;
        q=(struct node*)malloc(sizeof(struct node));
        while(c<pos)
        {
            q=p;
            p=p->link2;
            c++;
        }
        x->data=d;
        q->link2=x;
        x->link1=q;
        x->link2=p;
    }
    p=h;
    while(p!=NULL)
    {
        printf("%d ",p->data);
        p=p->link2;
    }
}
```

Output :

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
3
1 2 4 7 8 9
```

Your Output (stdout)

```
1 2 5 4 7 8 9
```

Expected Output

```
1 2 5 4 7 8 9
```

Q2. Delete an element from a given position in doubly linked list

Code :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *link1;
    struct node *link2;
};
int main()
{
    int i,pos,l;
    char a[50];
    scanf("%d",&pos);
    getchar();
    fgets(a,50,stdin);
    l=(strlen(a)/2);
    int ar[l];
    int dd=0,r;
    for(i=0;i<strlen(a);i++)
    {
        if((int)a[i]>=48)
        {
```

```
        r=(int)a[i]-48;
        ar[dd++]=r;
    }
}
l++;
struct node *p,*h;
p=h=(struct node*)malloc(sizeof(struct node));
p->link1=NULL;
p->data=ar[0];
for(i=1;i<l;i++)
{
    p->link2=(struct node*)malloc(sizeof(struct node));
    p=p->link2;
    p->data=ar[i];
}
p->link2=NULL;
p=h;
int c=1;
if(pos==1)
{
    p=p->link2;
    p->link1=NULL;
    free(h);
    h=p;
}
else if(pos>l)
{
    struct node *xx;
    xx=(struct node*)malloc(sizeof(struct node));
    while(p->link1!=NULL)
    {
        xx=p;
        p=p->link1;
    }
    xx->link2=NULL;
    free(p);
}
else
{
    struct node *q,*gh;
    q=(struct node*)malloc(sizeof(struct node));
    gh=(struct node*)malloc(sizeof(struct node));
    while(c<pos)
    {
        q=p;
        p=p->link2;
        gh=p->link2;
        c++;
    }
    q->link2=gh;
```

```
        gh->link1=q;
        free(p);
    }
    p=h;
    while(p!=NULL)
    {
        printf("%d ",p->data);
        p=p->link2;
    }
```

Output :

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
2
1 2 3 4 5
```

Your Output (stdout)

```
1 3 4 5
```

Expected Output

```
1 3 4 5
```

Q3. Insert at the beginning and at the end of a doubly linked list.

Code :

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *prev,*next;
};
struct node *CreateList(struct node *head,int a[],int l);
void DisplayList(struct node *head);
struct node *InsertBeginning(struct node *head,int beg);
struct node *InsertEnd(struct node *head,int end);
```

```
int main() {
    struct node *head=NULL;
    int i,beg,end,l;
    char a[50];
    scanf("%d",&beg);
    scanf("%d",&end);
    getchar();
    fgets(a,50,stdin);
    l=(strlen(a)/2);
    int ar[l];
    int dd=0,r;
    for(i=0;i<strlen(a);i++)
    {
        if((int)a[i]>=48)
        {
            r=(int)a[i]-48;
            ar[dd++]=r;
        }
    }
    l++;
    head=CreateList(head,ar,dd);
    head=InsertBeginning(head,beg);
    head=InsertEnd(head,end);
    DisplayList(head);
    return 0;
}
struct node *CreateList(struct node *head,int a[],int l)
{
    struct node *tmp,*newnode=(struct node*)malloc(sizeof(struct node));
    int i;
    if(newnode==NULL)
    {
        printf("Memory can not be allocated.");
        exit(0);
    }
    newnode->data=a[0];
    newnode->prev=NULL;
    newnode->next=NULL;
    head=newnode;
    tmp=head;
    for(i=1;i<l;i++)
    {
        struct node *newnode=(struct node*)malloc(sizeof(struct node));
        if(newnode==NULL)
        {
            printf("Memory can not be allocated.");
            exit(0);
        }
        newnode->data=a[i];
        newnode->next=NULL;
```

```
        tmp->next=newnode;
        newnode->prev=tmp;
        tmp=newnode;
    }
    return head;
}
void DisplayList(struct node *head)
{
    if(head==NULL)
    {
        printf("The list is empty.");
    }
    else
    {
        struct node *tmp=head;
        while(tmp!=NULL)
        {
            printf("%d ",tmp->data);
            tmp=tmp->next;
        }
    }
}
struct node *InsertBeginning(struct node *head,int beg)
{
    struct node *newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Memory can't be allocated.");
        exit(0);
    }
    newnode->data=beg;
    newnode->next=head;
    head->prev=newnode;
    newnode->prev=NULL;
    head=newnode;
    return head;
}
struct node *InsertEnd(struct node *head,int end)
{
    struct node *tmp=head,*newnode=(struct node*)malloc(sizeof(struct
node));
    if(newnode==NULL)
    {
        printf("Memory can't be allocated.");
        exit(0);
    }
    newnode->data=end;
    newnode->next=NULL;
    while(tmp->next!=NULL)
    {
```

```
        tmp=tmp->next;
    }
    tmp->next=newnode;
    newnode->prev=tmp;
    return head;
}
```

Output :

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
2
10
1 2 3 4 5
```

Your Output (stdout)

```
2 1 2 3 4 5 10
```

Expected Output

```
2 1 2 3 4 5 10
```

Q4. Insert elements at the end and the beginning of a circular linked list

Code :

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *prev,*next;
};
struct node *CreateList(struct node *head,int a[],int l);
void DisplayList(struct node *head);
struct node *InsertBeginning(struct node *head,int beg);
struct node *InsertEnd(struct node *head,int end);

int main() {
    struct node *head=NULL;
```

```
    int i,beg,end,l;
    char a[50];
    scanf("%d",&beg);
    scanf("%d",&end);
    getchar();
    fgets(a,50,stdin);
    l=(strlen(a)/2);
    int ar[l];
    int dd=0,r;
    for(i=0;i<strlen(a);i++)
    {
        if((int)a[i]>=48)
        {
            r=(int)a[i]-48;
            ar[dd++]=r;
        }
    }
    l++;
    head=CreateList(head,ar,dd);
    head=InsertBeginning(head,beg);
    head=InsertEnd(head,end);
    DisplayList(head);
    return 0;
}
struct node *CreateList(struct node *head,int a[],int l)
{
    struct node *tmp,*newnode=(struct node*)malloc(sizeof(struct node));
    int i;
    if(newnode==NULL)
    {
        printf("Memory can not be allocated.");
        exit(0);
    }
    newnode->data=a[0];
    newnode->prev=NULL;
    newnode->next=NULL;
    head=newnode;
    tmp=head;
    for(i=1;i<l;i++)
    {
        struct node *newnode=(struct node*)malloc(sizeof(struct node));
        if(newnode==NULL)
        {
            printf("Memory can not be allocated.");
            exit(0);
        }
        newnode->data=a[i];
        newnode->next=NULL;
        tmp->next=newnode;
        newnode->prev=tmp;
```

```
        tmp=newnode;
    }
    return head;
}
void DisplayList(struct node *head)
{
    if(head==NULL)
    {
        printf("The list is empty.");
    }
    else
    {
        struct node *tmp=head;
        while(tmp!=NULL)
        {
            printf("%d ",tmp->data);
            tmp=tmp->next;
        }
    }
}
struct node *InsertBeginning(struct node *head,int beg)
{
    struct node *newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Memory can't be allocated.");
        exit(0);
    }
    newnode->data=beg;
    newnode->next=head;
    head->prev=newnode;
    newnode->prev=NULL;
    head=newnode;
    return head;
}
struct node *InsertEnd(struct node *head,int end)
{
    struct node *tmp=head,*newnode=(struct node*)malloc(sizeof(struct
node));
    if(newnode==NULL)
    {
        printf("Memory can't be allocated.");
        exit(0);
    }
    newnode->data=end;
    newnode->next=NULL;
    while(tmp->next!=NULL)
    {
        tmp=tmp->next;
    }
```

```
    tmp->next=newnode;
    newnode->prev=tmp;
    return head;
}
```

Output :

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
3
4
1 2 3 4
```

Your Output (stdout)

```
3 1 2 3 4 4
```

Expected Output

```
3 1 2 3 4 4
```

Q5. delete an element from a given position in circular linked list

Code :

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

struct node
{
    int data;
    struct node *next,*prev;
};
void create(struct node **head,int n);
void display(struct node **head);
void delete(struct node **head,int pos);

int main()
{
    int pos,n;
    struct node *head=NULL;
    scanf("%d",&pos);
    n=pos+2;
```

```
    create(&head,n);
    delete(&head,pos);
    display(&head);
    return 0;
}
void create(struct node **head,int n)
{
    int i,data;
    struct node *prevnode,*newnode;
    prevnode=NULL;
    for(i=1;i<=n;i++)
    {
        newnode=(struct node *)malloc(sizeof(struct node));
        scanf("%d",&data);
        newnode->data=data;
        newnode->next=NULL;
        newnode->prev=NULL;
        if(prevnode!=NULL)
        {
            prevnode->next=newnode;
            newnode->prev=prevnode;
        }
        if(*head==NULL)
            *head=newnode;
        prevnode=newnode;
    }
    prevnode->next=*head;
}
void display(struct node **head)
{
    struct node *current;
    int n=1;
    if(*head==NULL)
    {
        printf("\n List is empty.");
        return;
    }
    current=*head;
    do
    {
        printf("%d ",current->data);
        current=current->next;
        n++;
    }while(current!=*head);
}
void delete(struct node **head,int pos)
{
    if(*head==NULL)
    {
        printf("\n List is empty.");
```

```
    }
    else
    {
        int c=1;
        struct node *x,*y;
        x=*head;
        y=NULL;
        while(c<pos)
        {
            y=x;
            x=x->next;
            c++;
        }
        y->next=x->next;
        x->next->prev=y;
        x->next=NULL;
        x->prev=NULL;
        free(x);
    }
}
```

Output :

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
1 7 3 5 3 7 10
```

Your Output (stdout)

```
1 7 3 5 7 10
```

Expected Output

```
1 7 3 5 7 10
```

Q6. Given two sorted doubly linked lists. Merge them into a single one without creating any new node.

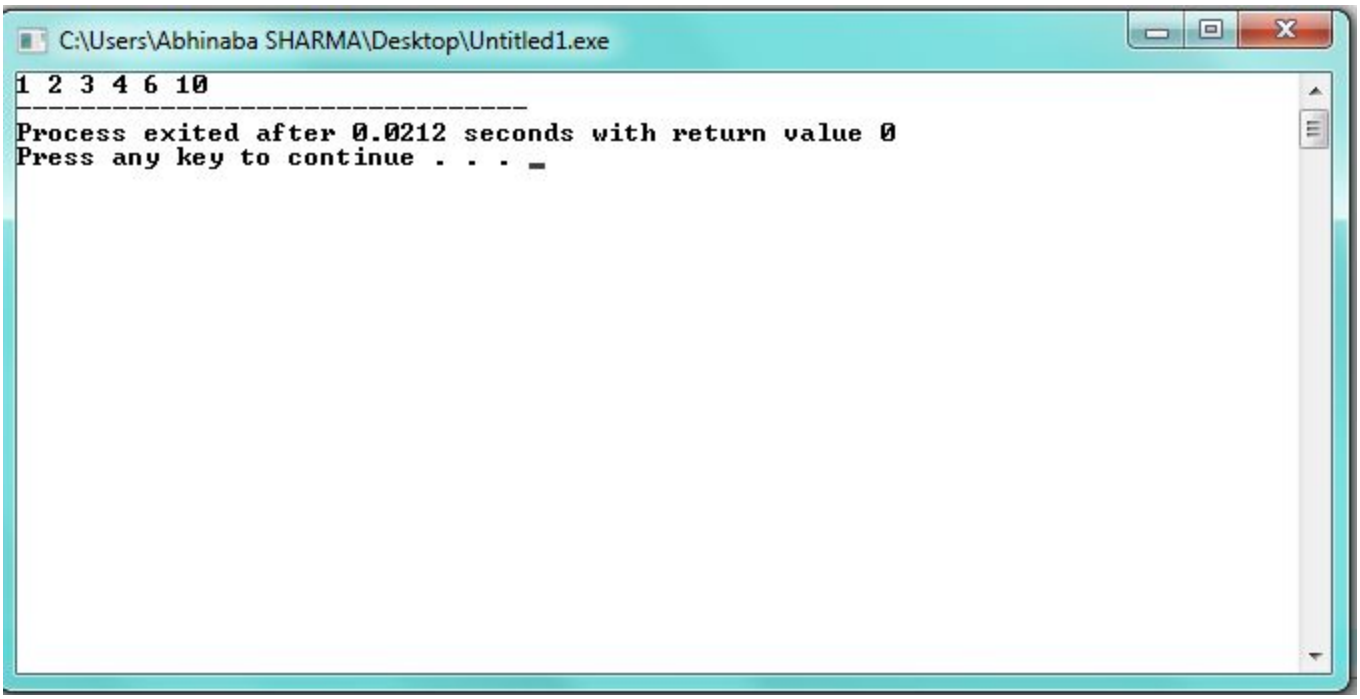
Code :

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


struct Node
{
    int data;
    struct Node* next;
};
void MoveNode(struct Node** destRef, struct Node** sourceRef);


struct Node* SortedMerge(struct Node* a, struct Node* b)
{

    struct Node dummy;


    struct Node* tail = &dummy;

    dummy.next = NULL;
    while (1)
    {
        if (a == NULL)
        {

            tail->next = b;
            break;
        }
        else if (b == NULL)
        {
            tail->next = a;
            break;
        }
        if (a->data <= b->data)
            MoveNode(&(tail->next), &a);
        else
            MoveNode(&(tail->next), &b);

        tail = tail->next;
    }
    return(dummy.next);
}


void MoveNode(struct Node** destRef, struct Node** sourceRef)
```

```
{

    struct Node* newNode = *sourceRef;
    assert(newNode != NULL);


    *sourceRef = newNode->next;


    newNode->next = *destRef;


    *destRef = newNode;
}



void push(struct Node** head_ref, int new_data)
{

    struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));


    new_node->data = new_data;


    new_node->next = (*head_ref);


    (*head_ref) = new_node;
}


void printList(struct Node *node)
{
    while (node!=NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}


int main()
{

    struct Node* res = NULL;
    struct Node* a = NULL;
    struct Node* b = NULL;
```

```
    push(&a, 6);
    push(&a, 4);
    push(&a, 3);

    push(&b, 10);
    push(&b, 2);
    push(&b, 1);


    res = SortedMerge(a, b);


    printList(res);

    return 0;
}
```

Output :



Q7. Reverse a circular linked list

Code :

```
#include "string.h"
#include<stdio.h>
#include <stdlib.h>
struct node
{
    int i;
    struct node *next;
    struct node *prev;
}*head=NULL,*pre;
```

```
void in(int l)
{
    scanf("%d",&l);
    struct node *temp=malloc(sizeof(struct node));

    if(head==NULL)
    {
        temp->i=l;
        temp->next=NULL;
        temp->prev=NULL;
         head=temp;
         pre=head;
    }else
    {
        temp->i=l;
        temp->next=NULL;
        pre->next=temp;
        temp->prev=pre;
        pre=pre->next;
    }
}
void show()
{
    struct node *temp=head;
    while(temp->next!=NULL)
    {
        temp=temp->next;
    }
    while(temp!=NULL)
    {
        printf("%d ",temp->i);
        temp=temp->prev;
    }
}
int main() {
    int j=0,no[100],k;
    char num[100];
    no[0]=0;
    fgets(num,100,stdin);
    for(int i=0;i<strlen(num);i++)
    {
        if(num[i]==' ')
        {
            j++;
            no[j]=0;
        }
        else
        {
            no[j]=no[j]*10+((int)num[i]-48);
        }
```

```
    }
    for(int i=0;i<=j;i++)
    {
        k=no[i];
        in(k);
    }
    show();
    return 0;
}
```

Output :

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
1 5 2 6 4
```

Your Output (stdout)

```
4 6 2 5 1
```

Expected Output

```
4 6 2 5 1
```

Q8. Given a circular linked list. Split it into two separate circular linked lists without creating new nodes. One will contain all the odd numbers. Another one will contain all the even numbers.

Code :

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

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

struct node *even=NULL;
```

```
struct node *odd=NULL;
struct node *ll=NULL;
void insert(int data)
{
    struct node *link,*present;
    link=(struct node *)malloc(sizeof(struct node));
    link->data=data;
    link->next=NULL;
    if(ll==NULL)
    {
        ll=link;
        ll->next=link;
        return;
    }
    present=ll;
    while(present->next!=ll)
        present=present->next;
    present->next=link;
    link->next=ll;
}
void display(struct node *h)
{
    struct node *ptr=h;
    while(ptr->next!=h)
    {
        printf("%d ",ptr->data);
        ptr=ptr->next;
    }
    printf("%d ",ptr->data);
}
void split()
{
    struct node *ll1;
    struct node *link;
    struct node *present;
    ll1=ll;
    while(ll1->next!=ll)
    {
        struct node *link = (struct node*) malloc(sizeof(struct node));
        link->data =ll1->data;
        link->next = NULL;
        if(ll1->data%2==0)
        {
            if(even==NULL)
            {
                even=link;
                even->next=link;
                ll1=ll1->next;
                continue;
            }
```

```
        else
        {
            present=even;
            while(present->next!=even)
            {
                present=present->next;
            }
            present->next=link;
            link->next=even;
        }
        ll1=ll1->next;
    }
    else
    {
        if(odd==NULL)
        {
            odd=link;
            odd->next=link;
            ll1=ll1->next;
            continue;
        }
        else
        {
            present=odd;
            while(present->next!=odd)
            {
                present=present->next;
            }
            present->next=link;
            link->next=odd;
        }
        ll1=ll1->next;
    }
}
link = (struct node*) malloc(sizeof(struct node));
link->data = ll1->data;
link->next = NULL;
if(ll1->data%2 == 0)
{
    present = even;
    while(present->next!=even)
    {
        present=present->next;
    }
    present->next=link;
    link->next=even;
}
else
{
    present=odd;
```

```
        while(present->next!=odd)
        {
            present=present->next;
        }
        present->next=link;
        link->next=odd;
    }
}
int main()
{
    int data,i;
    for(i=1;i<=7;i++)
    {
        scanf("%d",&data);
        insert(data);
        data=0;
    }
    split();
    display(odd);
    printf("\n");
    display(even);
    return 0;
}
```

Output :