## 1、

#include <stdio.h>

#include <stdlib.h>

#define M 1024

typedef struct stack

{

int d[M];

int top;

}S;

void Init(S \*s)

{

s->top=0;

}

int Empty(S \*s)

{

if(s->top>0)

return 0;

else

return 1;

}

int Full(S \*s)

{

if(s->top!=M)

return 0;

else

return 1;

}

void Push(S \*s, int x)

{

if(s->top==M)

printf("Overflow\n");

else

s->d[s->top++]=x;

}

int Pop(S \*s)

{

if(s->top==0)

return 0;

else

{

s->top--;

return s->d[s->top];

}

}

int Size(S \*s)

{

return (s->top);

}

int Top(S \*s)

{

if(s->top==0)

return 0;

else

return (s->d[s->top-1]);

}

int main()

{

S \*s;

s=(S\*)malloc(sizeof(S));

Init(s);

printf("%d\n",Empty(s));

Push(s,1);

printf("%d\n",Empty(s));

Pop(s);

printf("%d\n",Top(s));

printf("%d\n",Full(s));

return 0;

}

## 2、

#include <stdio.h>

#include <stdlib.h>

typedef struct link

{

int info;

struct link \*next;

}L;

typedef struct

{

L \*top;

}LS;

void Init(LS \*ls)

{

ls->top=NULL;

}

void Push(LS \*ls, int x)

{

L \*l=NULL;

l=(L\*)malloc(sizeof(L));

l->info=x;

l->next=ls->top;

ls->top=l;

}

int Pop(LS \*ls)

{

L \*p=NULL;

int x;

if(ls->top==NULL)

return 0;

else

{

x=(ls->top)->info;

p=ls->top;

ls->top=p->next;

free(p);

return x;

}

}

int Top(LS \*ls)

{

if(ls->top==NULL)

return 0;

else

return (ls->top)->info;

}

int Empty(LS \*ls)

{

if(ls->top==NULL)

return 1;

else

return 0;

}

int main()

{

LS \*ls;

ls=(LS\*)malloc(sizeof(LS));

Init(ls);

printf("%d\n",Empty(ls));

Push(ls,2);

printf("%d\n",Empty(ls));

printf("%d\n",Pop(ls));

printf("%d\n",Top(ls));

return 0;

}

## 3、

#include <stdio.h>

#include <stdlib.h>

#define m 1024

typedef struct

{

int d[m];

int f,r;

}Q;

void Init(Q \*q)

{

q->f=0;

q->r=0;

}

int Empty(Q \*q)

{

if(q->r==q->f)

return 1;

return 0;

}

int Full(Q \*q)

{

if((q->r+1)%m==q->f)

return 1;

else

return 0;

}

int Size(Q \*q)

{

return ((m+q->r-q->f)%m);

}

int Head(Q \*q)

{

if(q->f==q->r)

return 0;

else

return (q->d[(q->f+1)%m]);

}

void EQ(Q \*q, int x)

{

if((q->r+1)%m==q->f)

printf("Overflow\n");

else

{

q->r=(q->r+1)%m;

q->d[q->r]=x;

}

}

int DQ(Q \*q)

{

if(q->f==q->r)

return 0;

else

{

q->f=(q->f+1)%m;

return (q->d[q->f]);

}

}

int main()

{

Q \*q;

q=(Q\*)malloc(sizeof(Q));

Init(q);

printf("%d\n",Full(q));

printf("%d\n",Empty(q));

EQ(q,1);

printf("%d\n",Empty(q));

printf("%d\n",Head(q));

DQ(q);

printf("%d\n",Head(q));

return 0;

}

## 4、

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

int d;

struct node \*next;

}N;

typedef struct

{

N \*f;

N \*r;

}LQ;

void Init(LQ \*lq)

{

lq->f=(N\*)malloc(sizeof(N));

lq->f->next=NULL;

lq->r=lq->f;

}

int Empty(LQ \*lq)

{

if(lq->f==lq->r)

return 1;

return 0;

}

int Size(LQ \*lq)

{

int i=0;

N \*p=lq->f->next;

while(p)

{

i++;

p=p->next;

}

return i;

}

int Head(LQ \*lq)

{

if(lq->f==lq->r)

return 0;

else

return (lq->f->next->d);

}

void EQ(LQ \*lq, int x)

{

N \*s;

s=(N\*)malloc(sizeof(N));

s->d=x;

s->next=NULL;

lq->r->next=s;

lq->r=s;

}

int DQ(LQ \*lq)

{

int x;

N \*p;

if(lq->f==lq->r)

return 0;

else

{

p=lq->f->next;

lq->f->next=p->next;

if(p->next==NULL)

lq->r=lq->f;

x=p->d;

free(p);

return x;

}

}

int main()

{

LQ \*lq;

lq=(LQ\*)malloc(sizeof(N));

Init(lq);

printf("%d\n",Empty(lq));

EQ(lq,6);

printf("%d\n",Empty(lq));

printf("%d\n",Head(lq));

DQ(lq);

printf("%d\n",Empty(lq));

return 0;

}