Tamal Datta

1/06/2020

1) **WAP to insert new node at the beginning using double linked list.**

Soln:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct abc{
   int a;
   struct abc* p;
   struct abc* n;
}abc;
void add_beg(abc **p);
void print(abc* hp);
void main()
{
   char c;
   abc* hp=0;
   do{
   add_beg(&hp);
   printf("add new node?\n");
   scanf(" %c",&c);
   }while(c=='Y' || c=='y');

   printf("**********printing the dll**********\n");
   print(hp);
}

void add_beg(abc **hp)
{
   abc *temp=malloc(sizeof(abc));
   printf("enter number\n");
   scanf("%d",&(temp->a));
   temp->n=0;
   temp->p=0;

   if(*hp==0)
      *hp=temp;
      else
   {
      temp->n=*hp;
      (*hp)->p=temp;
      *hp=temp;
   }
}
```

```
void print(abc* hp)
{
 if(hp)
 {
    printf("\t%d\n",hp->a);
    print(hp->n);
 }
}
```

2) WAP to insert new node at the ending using double linked list.

Soln:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct abc{
   int a;
   struct abc* n;
   struct abc* p;
}abc;
void add_end(abc **hp);
void print(abc *hp);
void main()
{
   char op;
   abc *hp=0;
   do{
      add_end(&hp);
      printf("add more??\n");
      scanf(" %c",&op);
   }while(op=='Y' || op=='y');

   printf("----------------------\n");
   print(hp);
};

void add_end(abc **hp)
{
   abc *t=malloc(sizeof(abc));
   printf("enter the number\n");
   scanf("%d",&(t->a));
   t->n=0;
   t->p=0;

   if(*hp==0)
   {
      *hp=t;
```

```
        }
        else
        {
            abc *temp=*hp;
            while(temp->n!=0)
                temp=temp->n;

                t->p=temp;
                temp->n=t;
        }
    }

    void print(abc *hp)
    {
        if(hp)
        {
            printf("%d\n",hp->a);
            print(hp->n);
        }
    }
```

3) WAP to insert new node at the middle using double linked list.

Soln:

```
#include<iostream>
using namespace std;
typedef struct abc{
    int a;
    struct abc* p;
    struct abc* n;
}abc;
void add_end(abc **hp);
void print(abc* hp);
int len(abc *t);
void app(abc& t,int n,abc **hp);
int main()
{
    char c;
    abc *hp=0;
    do{
        add_end(&hp);
        cout<<"add more node?? y/n"<<endl;
        cin>>c;
    }while(c=='y');

    print(hp);
    cout<<"enter the position of new node(starting from position 1)"<<endl;
```

```
    int n;
    cin>>n;
    abc t;
    t.n=0;
    t.p=0;
    cout<<"enter value of new node"<<endl;
    cin>>t.a;
    app(t,n,&hp);
    cout<<"after appending new node at position"<<n<<endl;
    print(hp);
}

void add_end(abc **hp)
{
    abc *t=new abc;
    cout<<"enter the number"<<endl;
    cin>>t->a;
    t->n=0;
    t->p=0;

    if(!(*hp))
    {
        *hp=t;
    }
    else{
        abc *t1=*hp;
        while(t1->n!=0)
            t1=t1->n;
        t->p=t1;
        t1->n=t;
    }
}

void print(abc* hp)
{
    if(hp)
    {
        cout<<hp->a<<endl;
        print(hp->n);
    }
}

void app(abc& t,int n,abc **hp)
{
    if(n==1)
    {
```

```
        t.n=*hp;
        (*hp)->p=&t;
        *hp=&t;
    }
    else if(n>=len(*hp))
    {
        abc *temp=*hp;
        while(temp->n)
            temp=temp->n;

        temp->n=&t;
        t.p=temp;

    }
    else
    {
        abc *temp=*hp;
        int c=1;
        while(c!=n)
        {
            c++;
            temp=temp->n;
        }
    t.n=temp;
    t.p=temp->p;
    (t.p)->n=&t;
    (t.n)->p=&t;
    }
}

int len(abc *t)
{
int c=1;
while(t)
    {
        c++;
        t=t->n;
    }
return c;
}
```

4) WAP to count nodes using double linked list.

Soln:

```
#include<iostream>
using namespace std;
typedef struct abc{
   int a;
   struct abc* n;
   struct abc* p;
}abc;
void add_end(abc** hp);
void print(abc*hp);
int count(abc *hp);

int main()
{
   abc *hp=0;
   char op;
   do{
      add_end(&hp);
      cout<<"add more node?"<<endl;
      cin>>op;
   }while(op=='Y' || op=='y');

   print(hp);
   cout<<"no. of nodes are: "<<count(hp)<<endl;
}

void add_end(abc** hp)
{
   abc *temp=(abc*)malloc(sizeof(abc));
   cout<<"enter number"<<endl;
   cin>>temp->a;
   temp->p=0;
   temp->n=0;

   if(*hp)
   {
   abc *t=*hp;
   while(t->n)
      t=t->n;
   t->n=temp;
   }
   else
      *hp=temp;
}
```

```
void print(abc*hp)
{
   if(hp)
   {
      cout<<hp->a<<endl;
      print(hp->n);
   }
}

int count(abc *hp)
{
   int c=0;
   while(hp)
   {
      hp=hp->n;
      c++;
   }
   return c;
}
```

5)WAP to sort the nodes using double linked list.

Soln:

```
#include<iostream>
using namespace std;
typedef struct abc{
    int a;
    struct abc* p;
    struct abc* n;
}abc;

void add_end(abc **hp);
void print(abc* hp);
int len(abc *t);
void sort(abc** hp);
int main()
{
    char c;
    abc *hp=0;
    do{
       add_end(&hp);
       cout<<"add more node?? y/n"<<endl;
       cin>>c;
    }while(c=='y');

    print(hp);
    cout<<"-----------------"<<endl;
```

```
    cout<<"after sorting"<<endl;
    sort(&hp);
    print(hp);
}

void add_end(abc **hp)
{
    abc *t=new abc;
    cout<<"enter the number"<<endl;
    cin>>t->a;
    t->n=0;
    t->p=0;

    if(!(*hp))
    {
        *hp=t;
    }
    else{
        abc *t1=*hp;
        while(t1->n!=0)
            t1=t1->n;
        t->p=t1;
        t1->n=t;
    }
}

void print(abc* hp)
{
    if(hp)
    {
        cout<<hp->a<<endl;
        print(hp->n);
    }
}

void sort(abc **hp)
{
    abc *temp=*hp;
    int i=0,l=len(*hp);
    abc **arr;
    arr=new abc*[l];
    while(temp)
    {
        arr[i++]=temp;
        temp=temp->n;
    }
```

```
    for(int i=0;i<l-1;i++)
       for(int j=i+1;j<l;j++)
          if((arr[i]->a)>(arr[j]->a))
             {
                abc *t;
                t=arr[i],arr[i]=arr[j],arr[j]=t;
             }
    for(int i=0;i<l-1;i++)
      {
         arr[i]->n=arr[i+1];
         arr[i+1]->p=arr[i];
      }
    *hp=arr[0];
    arr[l-1]->n=0;
    arr[0]->p=0;

}

int len(abc *t)
{
int c=0;
while(t)
    {
       c++;
       t=t->n;
    }
return c;
}
```

6) WAP to reverse the nodes using double linked list.

Soln:

```
#include<iostream>
using namespace std;
typedef struct abc{
    int a;
    struct abc* p;
    struct abc* n;
}abc;
void add_end(abc **hp);
void print(abc* hp);
int len(abc *t);
void rev(abc** hp);
int main()
{
    char c;
```

```
    abc *hp=0;
    do{
        add_end(&hp);
        cout<<"add more node?? y/n"<<endl;
        cin>>c;
    }while(c=='y');

    print(hp);
    cout<<"after reversing"<<endl;
    rev(&hp);
    print(hp);
}

void add_end(abc **hp)
{
    abc *t=new abc;
    cout<<"enter the number"<<endl;
    cin>>t->a;
    t->n=0;
    t->p=0;

    if(!(*hp))
    {
        *hp=t;
    }
    else{
        abc *t1=*hp;
        while(t1->n!=0)
            t1=t1->n;
        t->p=t1;
        t1->n=t;
    }
}

void print(abc* hp)
{
    if(hp)
    {
        cout<<hp->a<<endl;
        print(hp->n);
    }
}

void rev(abc **hp)
{
    abc *t=*hp,*t1,*t2;
```

```
    while(t->n)
        t=t->n;
    (*hp)->n=0;
    *hp=t;
    t1=t->p;
    t->n=t1;
    t->p=0;
    t2=t;
    t=t1;
    while(t)
        {

            t1=t->p;
            t->n=t1;
            t->p=t2;
            t2=t;
            t=t1;
        }
}

int len(abc *t)
{
int c=0;
while(t)
    {
        c++;
        t=t->n;
    }
return c;
}
```

7) WAP to delete particular node using double linked list.

Soln:

```
#include<iostream>
using namespace std;
typedef struct abc{
    int a;
    struct abc* p;
    struct abc* n;
}abc;
void add_end(abc **hp);
void print(abc* hp);
void del(abc** hp,int n);
int main()
{
    char c;
```

```
    abc *hp=0;
    do{
        add_end(&hp);
        cout<<"add more node?? y/n"<<endl;
        cin>>c;
    }while(c=='y');

    print(hp);
    cout<<"enter the node number to be deleted(started from 1)"<<endl;
    int n;
    cin>>n;

    del(&hp,n);
    cout<<"after deleting node "<<n<<endl;
    print(hp);
}

void add_end(abc **hp)
{
    abc *t=new abc;
    cout<<"enter the number"<<endl;
    cin>>t->a;
    t->n=0;
    t->p=0;

    if(!(*hp))
    {
        *hp=t;
    }
    else{
        abc *t1=*hp;
        while(t1->n!=0)
            t1=t1->n;
        t->p=t1;
        t1->n=t;
    }
}

void print(abc* hp)
{
    if(hp)
    {
        cout<<hp->a<<endl;
        print(hp->n);
    }
}
```

```
void del(abc** hp,int n)
{
   int c=1;
   abc* t=*hp;
   while(c!=n)
      {
         t=t->n;
         c++;
      }
   if(n==1)
   {
      *hp=(*hp)->n;
      delete t;
   }
   else if(t->n==0)
   {
      t->p->n=0;
      delete t;
   }
   else
   {
      t->p->n=t->n;
      t->n->p=t->p;
      delete t;
   }
}
```

8) WAP to save nodes using double linked list.
Soln:

```
#include<iostream>
#include<fstream>
using namespace std;
typedef struct abc{
   int a;
   struct abc* n;
   struct abc* p;
}abc;
void add_end(abc** hp);
void print(abc*hp);
void save(const char*s,abc* hp);
int main()
{
   abc *hp=0;
   char op;
   do{
```

```
        add_end(&hp);
        cout<<"add more node?"<<endl;
        cin>>op;
    }while(op=='Y' || op=='y');

    print(hp);
    char s[10];
    cout<<"enter file name"<<endl;
    cin>>s;
    save(s,hp);
}

void add_end(abc** hp)
{
    abc *temp=new abc;
    cout<<"enter number"<<endl;
    cin>>temp->a;
    temp->p=0;
    temp->n=0;

    if(*hp)
    {
    abc *t=*hp;
    while(t->n)
        t=t->n;
    t->n=temp;
    }
    else
        *hp=temp;
}

void print(abc*hp)
{
    if(hp)
    {
        cout<<hp->a<<endl;
        print(hp->n);
    }
}

void save(const char*s,abc* hp)
{
    ofstream ofs(s);
    while(hp)
    {
        ofs<<hp->a<<endl;
```

```
        hp=hp->n;
    }
    ofs.close();
}
```

9) WAP to search particular node using double linked list.

Soln:

```
#include<iostream>
#include<fstream>
using namespace std;
typedef struct abc{
    int a;
    struct abc* n;
    struct abc* p;
}abc;
void add_end(abc** hp);
void print(abc*hp);
int find(int s,abc* hp);
int main()
{
    abc *hp=0;
    char op;
    do{
        add_end(&hp);
        cout<<"add more node?"<<endl;
        cin>>op;
    }while(op=='Y' || op=='y');

    print(hp);
    int s;
    cout<<"enter number to find"<<endl;
    cin>>s;
    int k=find(s,hp);
    k?cout<<"present at node "<<k<<endl:cout<<"node not present"<<endl;

}

void add_end(abc** hp)
{
    abc *temp=new abc;
    cout<<"enter number"<<endl;
    cin>>temp->a;
    temp->p=0;
    temp->n=0;
```

```
    if(*hp)
    {
    abc *t=*hp;
    while(t->n)
        t=t->n;
    t->n=temp;
    }
    else
        *hp=temp;
}

void print(abc*hp)
{
    if(hp)
    {
        cout<<hp->a<<endl;
        print(hp->n);
    }
}

int find(int s,abc* hp)
{
    int c=1;
    while(hp)
    {
        if(hp->a==s)
            return c;
        c++;
        hp=hp->n;
    }
    return 0;
}
```