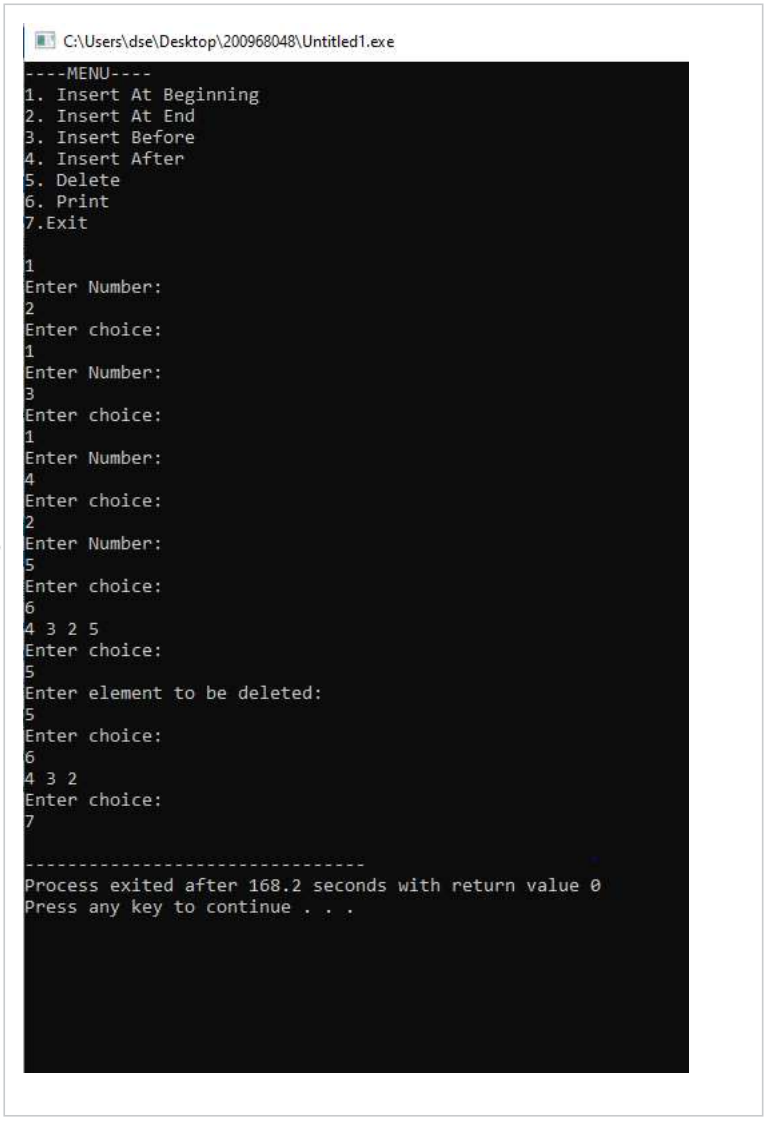
Week4 Batch3

Saturday, December 11, 2021 3:24 PM

1. Write a menu driven program to perform the following operations on linked list.
a) Insert an element in the beginning of the list
b) Insert an element at the end of the list
c) Insert an element before another element in the existing list
d) Insert an element after another element in the existing list
e) Delete a given element from the list
f) Print the list

```
#include<iostream>

using namespace std;

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

Node* head = NULL;

void InsertAtBeg(int x)
{
Node* temp = new Node();
if(head == NULL)
{
temp -> data = x;
temp -> next = head;
head = temp;
return;
}
else
{
temp -> data = x;
temp -> next = head;
head = temp;
}
}



void InsertAtEnd(int x)
{
if(head == NULL)
{
Node* g = new Node();
g -> data = x;
g -> next = head;
head = g;
}
Node* temp = head;
while(temp -> next != NULL)
{
temp = temp -> next;
```



```
}
Node* y = new Node();
y -> data = x;
y -> next = NULL;
temp -> next = y;
}

void InsertBefore(int ele , int x)
{
Node* t2 = head;
Node* ins = new Node();
Node* trav = head;
if(head == NULL)
{
cout<<"List is empty!\n";
return;
}
if(head -> data == ele)
{
ins -> data = x;
ins -> next = head;
head = ins;
return;
}

while(trav != NULL)
{
if(trav -> data == ele)
{
break;
}
trav = trav -> next;
}
if(trav == NULL)
{
cout<<"Element not found!\n";
return;
}
else
{
while(t2 -> next != trav)
{
t2 = t2 -> next;
}
ins -> data = x;
ins -> next = trav;
t2 -> next = ins;
}
}

void InsertAfter(int ele, int x)
{
if(head == NULL)
{
cout<<"List is empty!\n";
return;
}
Node* trav = head;
while(trav != NULL)
```

```
{
if(trav -> data == ele)
{
break;
}
trav = trav -> next;
}
if(trav == NULL)
{
cout<<"Element not found!\n";
return;
}
else
{
Node* ins  = new Node();
ins -> data = x;
ins -> next = trav -> next;
trav -> next = ins;
return;
}
}

void Delete(int x)
{
Node* temp = head;
Node* trav = head;
if(head == NULL)
{
cout<<"List is empty!\n";
return;
}
if(head -> data == x)
{
head = temp -> next;
delete(temp);
return;
}
else
{
while(temp != NULL)
{
if(temp -> data == x)
break;
temp = temp -> next;
}
if(temp == NULL)
{
cout<<"Element to be deleted not found!\n";
return;
}
else
{
while(trav -> next != temp )
{
trav = trav -> next;
}
trav -> next = temp -> next;
delete(temp);
return;
```

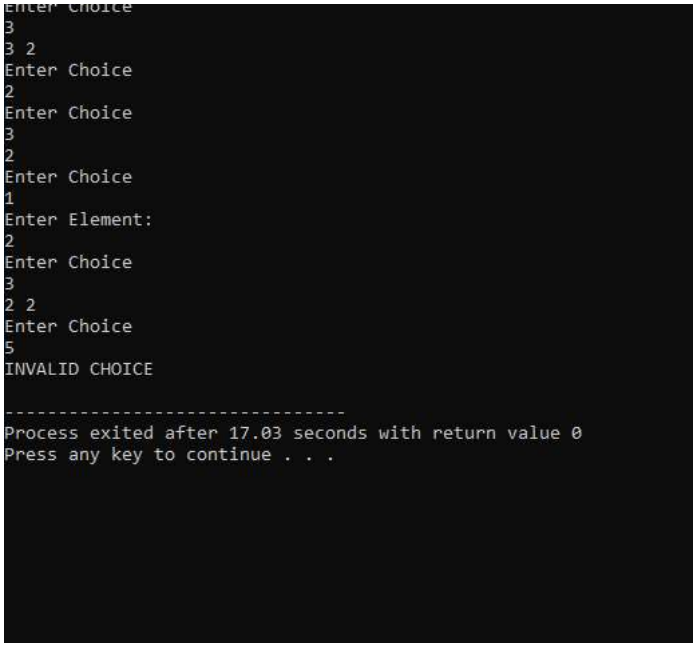
```
}
}
}

void Display()
{
if(head == NULL)
{
cout<<"List is empty!\n";
return;
}
Node* temp = head;
while(temp != NULL)
{
cout<<temp->data<<" ";
temp = temp -> next;
}
cout<<'\n';
}

int main()
{
cout << "----MENU----" << endl;
cout << "1. Insert At Beginning \n";
cout << "2. Insert At End \n";
cout << "3. Insert Before \n";
cout << "4. Insert After \n";
cout << "5. Delete  \n";
cout << "6. Print \n";
cout << "7.Exit \n";
int c;
cin>>c;
while(c != 7)
{
if(c == 1)
{
int e;
cout<<"Enter Number: \n";
cin>>e;
InsertAtBeg(e);
cout<<"Enter choice: \n";
cin>>c;
}
else if(c == 2)
{
int e;
cout<<"Enter Number: \n";
cin>>e;
InsertAtEnd(e);
cout<<"Enter choice: \n";
cin>>c;
}
else if(c == 3)
{
int pos;
int e;
cout<<"Enter the element before which you want to insert: \n";
cin>>pos;
cout<<"Enter Number: \n";
```

```
cin>>e;
InsertBefore(pos,e);
cout<<"Enter choice: \n";
cin>>c;
}
else if(c == 4)
{
int pos;
int e;
cout<<"Enter the element after which you want to insert: \n";
cin>>pos;
cout<<"Enter Number: \n";
cin>>e;
InsertAfter(pos,e);
cout<<"Enter choice: \n";
cin>>c;
}
else if(c == 5)
{
int d;
cout<<"Enter element to be deleted: \n";
cin>>d;
Delete(d);
cout<<"Enter choice: \n";
cin>>c;
}
else if(c == 6)
{
Display();
cout<<"Enter choice: \n";
cin>>c;
}
else
{
cout<<"Invalid choice!!\n";
break;
}
}
return 0;
}
```

2. Implement Stack and Queue using linked lists

```
#include <iostream>

using namespace std;

struct Node
{
int data;
struct Node *next;
}*top=NULL;
```

```
void push(int x)
{
struct Node *t;
t=new Node;

if(t==NULL)
printf("stack is full\n");
else
{
t->data=x;
t->next=top;
top=t;
}

}

int pop()
{
struct Node *t;
int x=-1;

if(top==NULL)
printf("Stack is Empty\n");
else
{
t=top;
top=top->next;
x=t->data;
delete(t);
}
return x;
}



void Display()
{
struct Node *p;
p=top;
while(p!=NULL)
{
printf("%d ",p->data);
p=p->next;
}
printf("\n");
}




struct qNode{
int qdata;
struct qNode *qnext;
}*front=NULL,*rear=NULL;


void enqueue(int x)
```



```
{
struct qNode *t;
t=new qNode;
if(t==NULL)
printf("Queue is FUll\n");
else
{
t->qdata=x;
t->qnext=NULL;
if(front==NULL)
front=rear=t;
else
{
rear->qnext=t;
rear=t;
}
}

}
int dequeue()
{
int x=-1;
struct qNode* t;
if(front==NULL)
cout<<"Queue is Empty"<<endl;
else
{
x=front->qdata;
t=front;
front=front->qnext;
delete(t);
}
return x;
}
void qDisplay()
{
struct qNode *p=front;
while(p)
{
printf("%d ",p->qdata);
p=p->qnext;
}
printf("\n");
}









int main()
{
int choice,ch,y,chq;
cout<<"1. Stack"<<endl;
cout<<"2. Queue"<<endl;
cin>>choice;
switch(choice)
```

```
{
case 1:
    cout<<"1.push"<<endl;
    cout<<"2.pop"<<endl;
    cout<<"3.display"<<endl;
    do{
    cout<<"Enter Choice"<<endl;
    cin>>ch;
    switch(ch){
       case 1:
          cout<<"Enter Element: "<<endl;
          cin>>y;
          push(y);
          break;
       case 2:
         pop();
         break;
       case 3:
         Display();
         break;
       default:
         cout<<"INVALID CHOICE"<<endl;
         exit(0);
         break;
    }
    }while(ch<=3);

case 2:
    cout<<"1.enqueue"<<endl;
    cout<<"2.dequeue"<<endl;
    cout<<"3.display"<<endl;
    do{
    cout<<"Enter Choice: "<<endl;
    cin>>chq;
    switch(chq){
       case 1:
          cout<<"Enter Element: "<<endl;
          cin>>y;
          enqueue(y);
          break;
       case 2:
         dequeue();
         break;
       case 3:
         qDisplay();
         break;
       default:
         cout<<"INVALID CHOICE"<<endl;
         break;
    }
}while(chq<=3);
}

return 0;
}
```