Page No.
Date 03/11/2022

Padmavati

1) Linked list :- inserting, deleting at rear & front end sorting, concat display.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *Node;
node getnode()
{
Node x;
x = (Node) malloc (sizeof (struct node));
if (x == NULL)
{
printf (" mem full \n");
exit (0);
}
return x;
}
void freenode (Node x)
{
free (x);
}
Node insert_front (Node first, int item)
{
Node temp;
temp = getnode();
```

```
temp->info=item;
temp->link= NULL;
 if (first == NULL)
return temp;
temp->link=first;
 first=temp;
return first;
}
node delete_front (Node first)
{
Node temp;
if (first == NULL)
{
printf ("list empty can't delete\n");
return first;
}
temp = first;
 temp= temp -> link
printf ("item deleted at front end is = %d\n",
            front->info);
free (first);
return temp;
}
Node insert_rear (Node first, int item)
{
Node temp, cur;
temp= getnode();
temp->info=item;
temp -> link = NULL;
```

```
if(first == NULL);
return temp;
cur = first;
while(cur->link != NULL)
cur = cur->link;
cur->link = temp;
return first;
}
Node deleterear(Node first)
{
Node cur, prev;
if(first == NULL)
{
printf("list empty")
return first;
}
if(first->link != NULL)
{
prev = cur;
cur = cur->link;
}
printf("item deleted at rear-end is %d",
cur->info);
free(cur);
prev->link = NULL;
return first;
}
Node order_list(int item, node first)
{
```

```
node temp, prev, cur;
temp = getnode();
temp->info = item;
temp->info = NULL;
if (first == NULL) return temp;
if (item < first->info)
{
temp->link = first;
return temp;
}
prev = NULL;
cur = first;
while (cur != NULL && item > cur->info)
{
prev = cur;
cur = cur->link;
}
prev->link = temp;
temp->link = cur;
return first;
}
node delete_info(int key, node first)
{
node prev, cur;
if (first == NULL)
{
printf ("list is empty\n");
return NULL;
}
```

```
if(key==first->info)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
while(cur!=NULL)
{
if(key==cur->info) break;
prev=cur;
cur=cur->link;
}
if(cur==NULL)
{
printf("search is so unsuccessful\n");
return first;
}
prev->link=cur->link;
printf("key deleted %d",cur->info);
freenode(cur);
return first;
}
void display(Node first)
{
node temp;
if(first==NULL)
```

```
printf ("list empty");
for (temp = first; temp != NULL; temp = temp->link
{
printf ("%d\n", temp->info);
}
}
void main ()
{
int item, choice, key;
Node first = NULL;
for (;;)
{
printf ("1: insert_front\n 2: delete_front\n
insert_rear\n 4: delete_rear\n 5: order_list\n 6:
delete_info\n 7: display\n 8: exit\n");
printf ("enter choice\n");
switch (choice)
{
case 1: printf ("enter the item at front_end\n");
scanf ("%d", &item);
first = insert_front (first, item);
break;
case 2: first = delete_front (first);
break;
case 3: printf ("enter the item at rear_end\n");
scanf ("%d", &item);
first = insert_rear (first, item);
break;
```

```
case 4 : first = delete_rear(first);
         break;
case 5 : printf("enter the item to be inserted
                 in at front\n");
         scanf("%d", &item);
         first = order_list(item, first);
         break;
case 6 : printf("enter the item to be deleted");
         scanf("%d", &key);
         first = delete_info(key, first);
         break;
case 7 : display(first);
         break;
default : exit(0);
break;
```

* linked list : concating & reversing

```
#include <stdio.h>
#include <conio.h>
struct node
{
int info;
struct node * link;
};
typedf struct node * Node;
node getnode()
{
node x;
x= (node) malloc (sizeof (struct node));
if (x == NULL)
{
printf ("mem full");
exit(0);
}
return x;
}
Node insert_rear (Node first, int item)
{
Node temp, cur;
temp = getnode();
temp->info = item;
temp->link = NULL;
if (first == NULL)
return temp;
cur = first;
```

```
while(cur->link != NULL)
cur = cur->link;
cur->link = temp;
return first;
}
Void display (Node first)
{
Node temp;
if (first == NULL)
printf ("list empty\n");
for(temp= first; temp!=NULL; temp= temp->link)
{
printf ("%d\n", temp->info);
}
Node concat(Node first, Node second)
{
Node cur;
if (first == NULL)
return second;
if (second == NULL)
return first;
cur = first;
while (cur->link != NULL)
cur = cur->link;
cur->link = second;
return first;
}
```

```
node reverse (node first)
{
node cur, temp;
cur = NULL;
while (first != NULL)
{
temp = first;
first = first -> link;
temp -> link = cur;
cur = temp;
}
return cur;
}
void main ()
{
int item, choice, pos, i, n;
node first = NULL, a, b;
for (;;)
{
printf (" 1. insert front \n 2. concat \n 3. reverse \n
 4. display \n 5. exit \n");
printf ("enter the choice \n");
scanf ("%d", &choice);
switch (choice)
{
case 1 : printf ("enter the item \n");
         scanf ("%d", &item);
         first = insert_rear (first, item);
         break;
```

```
case 2: printf("enter no of node in 1\n");
        scanf("%d", &n);
        a= NULL;
        for(i=0; i<n; i++)
        {
        printf("enter the item\n");
        scanf("%d", &item);
        a=insert_rear(a, item);
        }
printf("enter no. of node in 2\n");
        scanf("%d", &n);
        b= NULL;
        for(i=0; i<n; i++)
        {
        printf("enter the item\n");
        scanf("%d", &item);
        b=insert_rear(b, item);
        }
a=concat(a,b);
display(a);
break;
case 3: first=reverse(first);
        display(first);
        break;
case 4: display(first);
        break
default exit(0);
}
}
```