classmate
Date
Page

LAB 8 STACK AND QUEUE

```
#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;
}

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 is empty");
        printf ("Contents : \n");
    for (temp = first; temp != NULL; temp = temp->li
    {
        printf ("%d\n", temp->info);
    }
}
NODE sort (NODE first)
{
    int swapped;
    NODE ptr1;
    NODE lptr = NULL;
```

```
if (first == NULL)
    return NULL;
do
{
    swapped = 0;
    ptr1 = first;
    while (ptr1 -> link != lptr)
    {
        if (ptr1 -> info > ptr1 -> link -> info)
        {
            int temp = ptr1 -> info;
            ptr1 -> info = ptr1 -> link -> info;
            ptr1 -> link -> info = temp;
            swapped = 1;
        }
        ptr1 = ptr1 -> link;
    }
    lptr = ptr1;
} while (swapped);
}
NODE reverse (NODE first)
{
    NODE cur, temp;
    cur = NULL;
    while (first != NULL)
    {
        temp = first;
```

```
        first = first -> link;
        temp -> link = cur;
        cur = temp;
      }
    return cur;
  }
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 delete_front (NODE first)
{
    NODE temp;
    if (first == NULL)
    {
      printf ("list is empty \n");
      return first;
    }
```

```
    temp = first -> link;
    printf("deleted item at front = %d\n", first->info);
    free(first);
    return temp;
}
NODE delete_rear(NODE first)
{
    NODE cur, prev;
    if (first == NULL)
    {
        printf("list is empty\n");
        return first;
    }
    if (first -> link == NULL)
    {
        printf("only one item in list and deleted item =
                %d", first -> info);
        free(first);
        return NULL;
    }
    prev = NULL;
    cur = first;
    while (cur -> link != NULL)
    {
        prev = cur;
        cur = cur -> link;
    }
```

```
        printf("deleted item at rear = %d\n", curinfo);
        free(cur);
        prev->link = NULL;
        return first;
    }
}
void main()
{
    int item, choice, ch, n;
    NODE first = NULL, a, b;
    NODE stack_first = NULL, queue_first = NULL;
    for(;;)
    {
        printf("1. Insert_rear\n2. sorting\n3. display\n
            4. Concatinating 2 lists\n5. removing
            last\n6. stack implementation\n7. Queue
            implementation\n8. exit\n");
        printf("enter 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: sort(first);
                    display(first);
```

```
case 3: display (first);
        break;
case 4: printf ("Enter the no. of nodes in
                 list \n");
        scanf ("%d", &n);
        a = NULL;
        for (int i=0; i<n; i++)
        {
          printf ("Enter the item \n");
          scanf ("%d", &item);
          a = insert_rear (a, item);
        }
        printf ("Enter items of nodes in list 2\n");
        scanf ("%d", &n);
        b = NULL;
        for (int i=0; i<n; i++)
        {
          printf ("Enter the item");
          scanf ("%d", &item);
          b = insert_rear (b, item);
        }
        a = concat (a,b)
        display (a);
        break;
case 5: first = reverse (first);
        display (first);
```

```
case 6: printf ("stack\n");
        for(;;)
        {
          printf("\n1: Insert rear\n2: Delete
                 rear\n3: Display\n\n
                 Exit\n");
          printf("Enter the choice\n");
          scanf("%d", &ch);
          switch(ch)
          {
            case1: printf("Enter the item
                   at rear-end\n");
                   scanf("%d", &item);
                   first = insert_rear(first, it
                   break;
            case2: first = delete_rear(first);
                   break;
            case 3: display (first);
                   break;
            default: ch=0;
          }
          if (ch==0)
            break;
        }
        break;
```

```
case 7: printf("QUEUE \n");
        for(;;)
        {
          printf("\n1: Insert - rear \n2: Delete -
                  front \n3: Display - list \n4:
                  Exit\n");
          printf("Enter the choice\n");
          scanf("%d", &ch);
          switch(ch)
          {
            case 1: printf("Enter the item at
                            rear-end\n");
                    scanf("%d", &item);
                    first = insert_rear(first, item);
                    break;
            case 2: first = delete_front(first);
                    break;
            case 3: display(first);
                    break;
            default: ch=0;
          }
          if(ch==0)
            break;
        }
        break;
    default: exit(0)
  }
}
```