EAST WEST UNIVERSITY

Assignment (01)

Department of CSE
Course: CSE207 Data Structures
Course code: CSE207

Submitted by:

Name: Izazul Hoq Imran
ID : 2019-3-60-043

Submitted to:

Dr. Maheen Islam
Associate Professor
Department of Computer
Science & Engineering

Date: 13 April 2021

Ans: to the Q no: 1

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node NODE;
struct Node
{
    int data;
    struct Node *next;
};

void insert (NODE ** l3, int item)
{
    NODE *p *temp;
    temp = (NODE *) malloc (size of (NODE));
    temp -> data = item;
    temp -> next = NULL;
    if (*l3 == NULL)
    {
        *l3 = temp;
    }
    else
    {
        p = *l3;
        while (p -> next != NULL)
        {
            p = p -> next;
        {
        p -> next = temp;
    }
}
```

```
NODE *create (NODE *l1)
{
    NODE *temp, *q;
    int n=4, m, i;
    printf ("Enter Data in Node:");
     for (i=0; i<n; i++)
       {
         scanf ("%d ", &m);
         temp = (NODE *) malloc (sizeof (NODE));
         temp -> data = m;
         temp -> next = NULL;
          if (l1== NULL)
               {
                  l1 = temp;
               }
               else
                {
                   q = l1;
                    while (q -> next != NULL)
                     {
                         q = q -> next;
                     {
                      q -> next = temp;
                   {
       {
       return l1;
{
```

```
NODE * newLinkedList (NODE *l1, NODE *l2)
{
    NODE *P1 = l1, *P2 = l2;
    NODE *l3 = NULL;
    NODE * temp;
    while (P1 != NULL)
    {
        int cmax = ((P1->data < P2->data) ? P2->data : P1->data);
        if (l3 == NULL)
        {
            temp = (NODE *) malloc (sizeof (NODE));
            temp->data = cmax;
            temp->next = NULL;
            l3 = temp;
        }
        else
        {
            insert (&l3, cmax);
        }
        P1 = P1->next;
        P2 = P2->next;
    }
    return l3;
}
void print (NODE *L3)
{
    while (l3 != NULL)
    { printf ("%d \t", l3->data);
      l3 = l3->next;
```

```
Void main ()
{
    NODE *l3 = NULL, *l2 = NULL; *l1 = NULL;
        l1 = create (l1);
        l2 = create (l2);
        l3 = Newlinkedlist (l1, l2)
        print (l3);
{
```

Ans: to the Q no: 2

Q2: Expression given that ((A-B)*X)*(C*(D+E)/F)

solution;

Expression	output/P	Stack
(		((
(		(((
A	A	(((
-	A	(((-
B	AB	(((-
)	AB-	((
*	AB-	((*
X	AB-X	((*
)	AB-X*	(
*	AB-X*	(*
(	AB-X*	(*(
C	AB-X*C	(*(
*	AB-X*C	(*(*
(	AB-X*C	(*(*(
D	AB-X*CD	(*(*(
+	AB-X*CD	(*(*(+
E	AB-X*CDE	(*(*(+
)	AB-X*CDE	(*(*
/	AB-X*CDE	(*(/
F	AB-X*CDEF	(*(/

Expression	Postfix/output	Stack
)	AB-X*CDE+*F/	(*
)	AB-X*CDE+*F/*	

Postfix expression is : AB-X*CDE+*F/*

Ans to the Q no:3

```
#include <stdio.h>
#include <stdlib.h>
Struct node
  {
     int data;
     Struct node *next;
  } *head;
create (int n)
  {
    Struct node *newNode, *temp;
    int data, i;
    head = malloc (sizeof (struct node));
    if (head = NULL)
        { printf ("No data");
          exit (0);
        }
    scanf ("%d", &data);
       head -> data = data;
       head -> next = NULL;
       temp = head;
           for ( i=2; i<=n; i++)
           {
             newNode = malloc (sizeof (struct node));
             if (newNode == NULL)
              {
                exit (0);
```

```
else
    scanf ("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        temp->next = newNode;
        temp = temp->next;
    }
}

void display ()
{
    struct node *temp;
    if (head == NULL)
    { printf ("Empty");
      exit (0)
    }
    else
        temp = head;
    while (temp != NULL)
    {
        printf ("%d ", temp->data);
            temp = temp->next;
    }
}
```

```
int swap (struct node *list, int P1, int P2)
{
   struct node *N1, *N2, *Pr1, *Pr2, *temp;
   int max = (P1 > P2) ? P1 : P2;
   int totalN = count (list);
if ((P1 <= 0 || P1 > totalN) || (P2 <= 0 || P2 > totalN))
    {
    return -10;
    }
if (P1 == P2)
    {
    return 100;
    }
    i = 1;
    temp = list;
    Pr1 = NULL;
    Pr2 = NULL;
    while (temp != NULL && (i <= max))
        {
            if (i==P1-1)
                Pr1 = temp;
        else if (i == P1)
                N1 = temp;
        else if (i == P2-1)
                Pr2 = temp
        else if (i == P2)
                N2 = temp;
```

```
    temp = temp -> next;
    i++;
  }
if (N1 != NULL && N2 != NULL)
  {
    if (Ptr1 != NULL)
      {
        Ptr1 -> next = N2;
      {
    if (Ptr2 != NULL)
        { Ptr2 -> next = N1;
        {
    temp = N1 -> next;
    N2 -> next = N1 -> next;
    N2 -> next = temp;
      if (Ptr1 == NULL)
          { head = N2;
          }
      else if (Ptr2 = NULL)
          { head = N1;
  }
}
int count (struct node * list)
    { int a=0;
```

```
while (list != NULL)
    {
      a++;
      list = list->next;
    {
return a;
}

int main()
  {
    int n, P1 = 1, P2;
    printf("How many data entry:");
    scanf("%d", &n);
    create(n);
    if (n/2 == 0)
      { printf("This node have no middle position");
        exit(0);
      }
    else
      P2 = (n/2)+1;
    if (swap(head, P1, P2) == 1)
      {
        printf("After swap:");
        display();
      }
  {
```

Ans: to the Q no:9

```
#include <stdio.h>
  Struct node {
        int data;
        Struct node * P, *next;
  {
Struct node *head;
Struct node * tail = NULL;

 void creatlist (int n)
   {
    Struct node * newnode, *temp;
     int data, i;
   Struct node *newNode = (Struct node*) malloc
              (sizeof (Struct node));
    if (head == NULL)
      { printf ("Here is no data:");
        exit (0);
       }
  Printf ("Enter data in Node : ");
      Scanf ("%d ", & data);
          head → data = data;
          head → next = NULL;
          temp = head;
                                    }
```

```
for (i=2; i<=n; i++)
    {
      newNode malloc (sizeof (struct node));
      if (newNode == '\0')
          { exit (0);
          }
      else
          newNode -> data = data;
          newNode -> next = NULL;
          temp -> next = newNode;
          temp = temp -> next;
    }
}
int rotate (int n)
  {
    struct node *c = head;
      if (int check == 0 || n >= check)
          { return;
          {
      else
          {
              for (int x = 1; x < n; x++)
                  {
                      c = c -> next;
                      head = c -> next;
```

```
    head -> p = NULL;
    tail = c;
    tail -> next = NULL;
  }
}

Void print ()
 {
    Struct node *c = head;
      if (head == NULL)
        { printf ("Empty ")
          exit (0);
        }
     while (c != NULL)
        { printf ("%d ", c->data);
          c = c->next;
        }
}

int main ()
      { int n, r;
        printf (" How much data entry ? : ");
        scanf ("%d", &n);
          create (n);
        printf (" counter clockwise Node no ? ");
        scanf ("%d", &r);
        rotate (r);
        Printf (" After rotate : ");
```

Ans: to the Q no: 7

```
#include <stdio.h>
  int a;
Struct Node
 {  int data;
    Struct Node *next;
 {* fornt = NULL ; *rear = NULL;

Void insert (int x)
  {
    Struct Node *newNode;
    int value;
    newNode = (Struct Node *) malloc (sizeof (Struct Node));
    for ( int i = 0; i<n; i++)
      {
        Scanf ("%d ", & value);
        newNode→ data = value;
        newNode → Next = NULL;
        if (front == NULL)
          fornt = rear = newNode;
        else
          {
            rarce → next = newNode;
            rear = newNode;
            a++;
          {
      {
}
```

```
void reverse ( int k , struct Node * stack )
{
  if (stack != NULL || k > a)
    {
    return ;
  {
  if (k <= 0)
    return
Node * temp;
  temp = (Node *) malloc (sizeof (struct Node));
    for (int i = 0; i < k ; i++)
     {
      *temp = fornt ;
       fornt = fornt -> next;
       free (temp)
     }
while (stack != NULL)
     {
      *stack = raer;
       free (stack);
     }
for (int i = a-k ; i <= 0; i--)
     {
      *stack = temp;
       temp = temp -> next;
       free ( temp);
     }
```

```
void print()
{ while (queue != NULL)
    {
      printf ("%d", queue->data);
      queue = queue->next;
    }

int main()
  {
    int k,n;
    printf ("How much data entry 1");
    scanf ("%d", &n);
    insert (n)
    scanf ("%d", &k);
    reverse (k);
    printf ("After reverse");
    print()
  }
```

Ans: to the Q no: 6

```
#include <stdio.h>
# include <stdlib.h>
  void Initialize (Queue *);
  int Empty (Queue *);
  int full (Queue*);
  void insert (item, Queue*);
   int empty1 (stack1*);
   int empty2 (stack2*);
    void push (stack1*, item);
    void enqueue (stack1*, stack2*);
     void pop (stack2*, item);
int item;
struct stack
 {
   itemtype item;
     struct stack *L1;
  }
  struct {
        stack *list1;
      } stack1;
  struct {
        stack *List2;
      } stack2;
```

```
struct
{
  stack1 input;
  stack2 output;
} queue;

int main()
{
  int i, j;
  Queue Q;
  Initialize(&Q);
    for (i=0; i<10; i++)
      {
        insert (i, &Q);
      }
   while (Q != NULL)
       {
         Remove (&Q, &j);
       }
}

void Initialize (Queue *Q)
   {
     Q -> input . List1 = NULL;
     Q -> output . List2 = NULL;
   }
```

```
int empty1 (stack1 * s)
  {
    return  s -> list1 == NULL;
  }
int empty2 (stack2 * s)
  {
    return s -> list2 == NULL;
  }
int empty (Queue *Q)
    {
      return (Empty1 (&Q -> input) && (empty
                    (&Q -> output)));
int Full (Queue *Q)
      {
        exit(0);
      }
void pop (stack2 * s, item * x)
    {
      stack *temp;
if( s -> list2 == NULL)
    {printf ("Empty list ");
    }
```

```
else
    {
      temp = S->list2;
      *x = temp->item;
      S->list2 = temp->L1;
      free (temp);
    }
}
void push ( stack_1 *S , item x)
    {
      Stack *temp;
      temp = (stack *) malloc (sizeof (stack));
       if (temp == NULL)
          { printf ("Error");
          }
        else
          {
            temp->L1 = S->list1;
            temp->item = x ;
            S->list1 = temp;
          }
    }
```

```
void Insert (itemT R, Queue *Q)
  {
   push (&(Q->input), R);
  }
void reverse (stack1 *s, stack2 *x)
  {
    Stack *temp;
    temp = s->list1->L1;
    s->list1->L1 = x->list2;
    x->list2 = s->list1;
    s->list1 = temp;
  }
void enqueue (stack1 *in, stack2 *out)
  {
    if (empty2(out))
    while (!empty1(in))
      {
        reverse (in, out)
      }
  }
void remove (Queue *Q, itemT *F)
  {
    enqueue (&(Q->input), &(Q->output));
    pop (&(Q->output, F);
  }
```

Ansi to the Q no: 5

```
#include <stdio.h>
#include <stdlib.h>
struct stack
{
  int data;
  struct stack *next;
} *top;
int x=0, y=0, z=0;
FILE *fp *ft;
void push (int n)
{
  int value;
  if (size >= c)
  {
    printf ("Stack overflow");
    return;
  }
struct stack * Newnode;
Newnode = (struct stack *) malloc (sizeof (
                                  . stack));
```

```
fp = fopen ("datastorce.txt", "a");

for( int i=0; i<n; i++)
  {
    newNode -> data = valuee;
    newnode -> next = top;
    top = newnode;
    x++;
  }
{

Void display ()
  {
    int i;
   if (s.top == -1)
    {
      printf ("stack empty");
    }
  else
    for (i = s.top ; i<=0; i--)
      {
        fp = fopen ("datastorce.txt", "r");
      while (fread (&newnode, sizeof (newnode)
            1, fp) == 1)
```

```
if (stack->data < 0)
    {
      temp = stack->data;
      pop (temp);
    }
else
   printf ("%d", stack->data);
      temp = stack->next;
    yy++;
  {
}

int pop ()
    {
      int data = 0;
      struct stack * topnode;
      if (size <= 0 || ! top)
          {
            printf ("Empty");
          }
      else
            for (int i=y ; i<=0 ; i--)
```

```
if (y<5)
  { printf ("Error!");
  }
else
    topnode = top;
    data = top->data;
    top = top->data;
    free (topnode)
    x--;
    return data;
}

int main()
    {
      int c, data, n;
      while (1)
        { printf ("1.write/push \n 2. Read ");
        switch (c)
         { case 1:
             scanf ("%d", &n);
             push (n)
             break;
```

```
case 2:
    Display();

default:
    printf("Invalid data entry");
    {
}
```

Ans: to the Q no: 8

```
solution;
    Q1 = createQueue
    S1 = createstack
    loop (not end of file)
    read number
      if (number not 0)
      pushstack (S1, number)
      else
        popstack (S1, x)
        popstack (S1, x)
        loop (not empty S1)
          pop stack (S1, x)
            enqueue (Q1, x)
          end loop
      end if
    end loop
```

Solution; popping out two numbers in stack the loop till empty numbers are 5, 7.

⇒ Before the "if" condition data are

5 23 34 67 8.6 4

⇒ After 0 has encountered else condition popping out two number in stack and executing loop numbers are

4, 6, 8, 67, 34, 5, 7

⇒ Read number before if condition

6 22 33 44

⇒ After 0 else condition till the stack empty

44, 33, 4, 6, 8, 67, 34, 5, 7

rear

front