```
Week -3
    Implementation of winder dunce
1 Include < Stdio. h>
# degine en SIZE 5
 int itums (SJZE)
 In+ toont =-1, rear =-1;
 int is futh () & I show mot while
  It (chont==rcont+1)11/lfront==0 gg
      rear == (stz==11))
   roturn 1;
                     mue ig 42
   return 0;
               7: mon hom som
int is Empty E) of bodies interval
  it (thont == -1) is sinds man with
    return 1;
    return o'
 Void En Owere (int rement) ]
  if (4 Fun (1)
  Printt ("In Queue is ful!! (n"):
 Eur &
 9+ (front = = -1) &
   front =0;
 oscur = (run +1) / SIZE;
  Hem [run] = Element;
```

```
Print of (" In insorted -> 1. d", Rement);
  3
                   1 Haman 1971 49
      de Quenc CPS
                            1 2019 .
   int ilement;
  if (yEmptyl) &.
   Print +: l'' in Ounce is Empty!! (n");
schwin (-1);
             and all a fused all Alog
  3 sur E.
   Clement = items [tront];
     1+ (hon +== rear) }
     front = -1;
   3 Eures
  funt = (bont +1) 1. STRE.
                       e ) was it
Print+ ("In Delete Generity ->"). of In"
    Element);
 ordern (denent).
                      En Jume ( ");
7
```

```
Void distray () L
   int 1;
   it (is Empty ())
    Privit (" in empty Onuce \n");
   sur &
 Print + ("In Front -) d'il front),
     drint+ (" In Itung ->12");+ + min
for (i= front; 1! = ron; i= (i+1) 1. SIZEY
    Print + ("" d" Herry CTD);
 Printt ("Kd", Hurry [T]);
Print ("In Pron -> y.dln", reor).
                           7. see 5
 3
             front = (mant +1) 1.5.
int main (25
 en Ource (17) ment melet 11/11 Formil
 En Over (2);
 en Queue (3),
                  e lune (duncot);
En anne (4);
En Queue (5);
 dist by ();
```

deaune (6); dedune (7); distay (); in a literal to the start of return o', · · · · · · b. d. i. 11. shoul hand. Inwested -) (Output Front > 2 Item\$ -> 3 4 5 6 7. Ron -> 1 : () foldish how Voct insulations Calculation how Coron - treating blow isty kohon bush (1104 3 hood) 41 Million History

2

```
Week -3
      Insultion Operation on
            Linked left i (1) running is
# include < stdio. h>
                          · O muij .
Hindude 2stdlib.hz
 Struct node
                Delput Invented -) 6
    int data;
    Street mode * next)
   3; 3 3 2 1 E . - ZIMHT
 Void desplay ();
 Void insut-bugin ();
 Void insurt - End(?
 void insert - Posco:
Struct node & Ptr;
 it (head == NULL)
     Print + ("list is Empty In");
    rown;
   8
  Pto = head;
  While (Arri = NULL)
```

E

Z

Elee

L

```
Orint + (".1.d In", Ptr -> data):
    Pto - Pto -> next:
           ( 10001 - 0 . (1) 11
   4
3
   iment - begins).
L
Struct node* temp;
temp = (Smeet rode *) malloc
 (Sizeg (Strut nøde));
 Print f l'enter the value to be
        Entered (n"); %
 Sion + ("i'd", & fumb -) data);
 temb -) rest = NULLIZ
 if ( hood stemb);
  it (head = NULL)
  head = tent;
  rhe &
     tumb -> rest = head;
    head = temp;
    John Snot or strong our
 Struct rode & tumb, & Ptr ;
tunt = (Struct mode 1*) malio c
Csize of (struct mode));
Printt (" Entor the value to be
           pensented (n");
```

```
temb-) next = NULL;
              It (head = NULL)
              E
                head = fun by
             3
             Elge
         ptr'= head;
Dhile (ptr - Snest != NULL)
                                2 "1 1 + n; A)
             & : ("m/ bo)
    (ore) (Artisper 1) + nois
            Par -> nest = temping ) = 3;
                         head . how!
      vord invit- Pusco
      3
        int Pos, 1;
       Some node a temb, * Ptr';
       Print of 1" enter the Position";
      Sant ("1-d", Bros),

temp = (9met node ") mollo c (size o)

Copiet mode 12
Print & C "Enter the Value top be
privated \n");
```

Switch (choice) (0) 2041 11 core !! inert -busines; breede; Troop : had Cose 2: inent-mal(); 1. 2. (brieflian = (H), (0-1) was 3', inut rose)) Brok : (3001 (- 24) - 240 Cose h; display () ; = then (- the) purols? = 1 in (- don't cose &: buil- (0); c) han byou brook; dyault; Print of (vinvaled Choice In"); break ; (11)/184 responsible to the state of 1111) 1 sing

```
1 most at Beginning
  angent at End
  enout at position
  Print light dist buy
5 suit
Enter your choice: 1
 Enter data to injert at beginning: 35
1. Ensert ad Beginning
2 Insurt of End
2 zonut at position
  2 Hay
  Socit
 Enter your choice; 3
 Enter data to inject at Beginning: 37
 Enter the Position to insert at: 2
1. Inent at Beginning
2 sowert at End
3 zuret at position
4. distley walt
enter your choice 2
Enter dather to dutet invent at End: 15
. Invert at Beginnling
2. Inevent at End
3. Quest at Position
4. display
    Sit
 Enter your Choice: 4
  linked list: 35 57
                                 15
 Enter your Choice: 5
             Sxit
```

```
#include <stdio.h>
#define SIZE 5
int items[SIZE];
int front = -1, rear = -1;
int isFull() {
    if ((front == rear + 1) || (front == 0 \&\& rear == SIZE - 1))
        return 1;
    return 0;
int isEmpty() {
    if (front == -1)
        return 1;
    return 0;
void enQueue(int element) {
    if (isFull())
        printf("\n Queue is full!! \n");
    else {
        if (front == -1)
            front = 0;
            rear = (rear + 1) \% SIZE;
            items[rear] = element;
            printf("\n Inserted->%d", element);
        } else {
            rear = (rear + 1) \% SIZE;
            items[rear] = element;
            printf("\n Inserted->%d", element);
```

```
int deQueue() {
    int element;
    if (isEmpty()) {
       printf("in Queue is empty !! \n");
        return (-1);
   } else {
        element = items[front];
        if (front == rear) {
            front = -1;
            rear = -1;
       } else {
            front = (front + 1) % SIZE;
        printf("\n Deleted elements ->%d \n", element);
        return (element);
void display() {
    int i;
    if (isEmpty())
       printf(" in Empty Queue \n");
    else {
        printf("\n Front->%d", front);
       printf("\n Items->");
        for (i = front; i = rear; i = (i + 1) \% SIZE) {
            printf("%d ", items[i]);
       printf("%d", items[i]);
       printf("\n Rear->%d\n", rear);
```

```
else {
        printf("\n Front->%d", front);
        printf("\n Items->");
        for (i = front; i != rear; i = (i + 1) % SIZE) {
            printf("%d ", items[i]);
        printf("%d", items[i]);
        printf("\n Rear->%d\n", rear);
int main() {
    enQueue(1);
    enQueue(2);
    enQueue(3);
    enQueue(4);
    enQueue(5);
    display();
    deQueue();
    deQueue();
    display();
    enQueue(6);
    enQueue(7);
    display();
    return 0;
```

```
Inserted->1
Inserted->2
Inserted->3
Inserted->4
Inserted->5
Front->0
Items->1 2 3 4 5
Rear->4
Deleted elements ->1
Deleted elements ->2
Front->2
Items->3 4 5
Rear->4
Inserted->6
Inserted->7
Front->2
Items->3 4 5 6 7
```

Rear->1

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
void insertAtBeginning(struct Node** head, int newData) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    newNode \rightarrow data = newData;
    newNode->next = *head;
    *head = newNode;
void insertAfter(struct Node* prevNode, int newData) {
    if (prevNode == NULL) {
        printf("Previous node cannot be NULL\n");
        return:
```

```
return;
    struct Node* lastNode = *head;
    while (lastNode->next != NULL) {
        lastNode = lastNode->next;
    lastNode->next = newNode;
void insertAtPosition(struct Node** head, int newData, int position) {
    if (position < 1) {
        printf("Invalid position\n");
        return;
    if (position == 1) {
        insertAtBeginning(head, newData);
        return;
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
```

```
newNode->data = newData;
    struct Node* temp = *head;
    for (int i = 1; i < position - 1 && temp != NULL; <math>i++) {
        temp = temp->next;
    if (temp == NULL) {
        printf("Position exceeds the length of the list\n");
        free(newNode);
        return;
    newNode->next = temp->next;
    temp->next = newNode;
void printList(struct Node* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
   printf("\n");
int main() {
    struct Node* head = NULL;
```

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    newNode \rightarrow data = newData;
    newNode->next = prevNode->next;
    prevNode->next = newNode;
void insertAtEnd(struct Node** head, int newData) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    newNode \rightarrow data = newData;
    newNode->next = NULL;
    if (*head == NULL) {
        *head = newNode;
```

```
int choice, data, position;
while (1) {
    printf("1. Insert at Beginning\n");
    printf("2. Insert at End\n");
    printf("3. Insert at Position\n");
    printf("4. Print List\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            printf("Enter data to insert at the beginning: ");
            scanf("%d", &data);
            insertAtBeginning(&head, data);
            break;
        case 2:
            printf("Enter data to insert at the end: ");
            scanf("%d", &data);
            insertAtEnd(&head, data);
            break:
        case 3:
            printf("Enter data to insert: ");
            scanf("%d", &data);
            printf("Enter position to insert at: ");
            scanf("%d", &position);
            insertAtPosition(&head, data, position);
            break:
        case 4:
            printf("Linked list: ");
            printList(head);
            break:
        case 5:
```

```
Lines oftenegrinizing
            break:
        case 2:
            printf("Enter data to insert at the end: ");
            scanf("%d", &data);
            insertAtEnd(&head, data);
            break:
        case 3:
            printf("Enter data to insert: ");
            scanf("%d", &data);
            printf("Enter position to insert at: ");
            scanf("%d", &position);
            insertAtPosition(&head, data, position);
            break:
        case 4:
           printf("Linked list: ");
            printList(head);
            break;
        case 5:
            while (head != NULL) {
                struct Node* temp = head;
                head = head->next;
                free(temp);
            return 0:
        default:
           printf("Invalid choice\n");
return 0;
```

```
    Insert at Beginning

Insert at End
Insert at Position
4. Print List
5. Exit
Enter your choice: 1
Enter data to insert at the beginning: 35

    Insert at Beginning

Insert at End
Insert at Position
4. Print List
5. Exit
Enter your choice: 3
Enter data to insert: 57
Enter position to insert at: 2

    Insert at Beginning

Insert at End
Insert at Position
4. Print List
5. Exit
Enter your choice: 2
Enter data to insert at the end: 15

    Insert at Beginning

Insert at End
Insert at Position

    Print List

5. Exit
Enter your choice: 4
Linked list: 35 57 15

    Insert at Beginning

Insert at End
Insert at Position
4. Print List
Exit
Enter your choice: 5
```