DS - Lab - 9

→ *) Pseudocode :-

Header files

define structure node with members info and link field

int info, struct node *link

Use typedef struct node *NODE

Function getnode()

→ Create node x

→ Allocate memory using malloc

→ if x==NULL, print memory full

→ else return x

Function freenode()

→ free the passed node

Function insert = front()

create NODE temp

temp = getnode() // allocate memory

put item in info field of temp

if list empty return temp

else temp → link = first

first = temp

return first

Function insert - rear ()

create 2 nodes temp and cur

allocate mem for temp temp = getnode ()

put item in temp → info

if list empty return temp

cur = first

move cur pointer to last position

cur → link = temp // place at rear end

return first

Function insert - pos ()

create 3 nodes temp, prev and cur

allocate memory for temp

put item in temp → info

if first == NULL and pos == 1

return temp

if first == NULL, return first

if pos == 1, first placed in temp → link

return temp

move to required position

prev → link ⇒ temp

temp → link = cur

return first

Function display ()

create node temp

print all elements

Function main()

for loop(;;)

give choices for all function do required operations.

*) Main code :-

```
#include <stdio.h>
#include <conio.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 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 insert_pos (int item, int pos, NODE first)
{
    NODE temp;
    NODE prev, cur;
    int count;
    temp = getnode ();
    temp -> info = item;
    temp -> link = NULL;
    if (first == NULL && pos == 1)
        return temp;
    if (first == NULL)
    {
        printf ("Invalid pos \n");
        return first;
    }
    if (pos == 1)
    {
        temp -> link = first;
        return temp;
```

```
        count = 1;
        prev = NULL;
        cur = first;
    while (cur != NULL && count != pos)
    {
        prev = cur;
        cur = cur -> link;
        count++;
    }
    if (count == pos)
    {
        prev -> link = temp;
        temp -> link = cur;
        return first;
    }
    printf ("IP\n");
    return first;
}
void display (NODE first)
{
    NODE temp;
    if (first == NULL)
printf ("list empty cannot display items\n");
```

```
for (temp = first; temp != NULL; temp =
                                temp -> link)
{
    printf("%d\n", temp -> info);
}
}
void main()
{
    int item, choice, pos;
    NODE first = NULL;
    for (;;)
{
    printf("\n1. Insert_front\n2: Insert_rear\n
3: Insert_pos\n4:- display_list\n5: Exit\n");
    printf("Enter the choice\n");
    scanf("%d", &choice);
    switch (choice)
    {
      case 1:
          printf("Enter the item at front-end\n");
          scanf("%d", &item);
          first = insert_front(first, item);
          break;
```

```
case 2:
    printf("Enter the item at rear-end\n");
    scanf("%d", &item);
    first = insert_rear(first, item);
    break;

case 3:
    printf("Enter the position and item :\n");
    scanf("%d", &pos);
    scanf("%d", &item);
    first = insert_pos(item, pos, first);
    break;

case 4:
    display(first);
    break;
default:
    exit(0);
}
}
}
```