<u>LAB PROGRAM 5 (SINGLY Linked List INSERT FRONT,REAR AND AT GIVEN POS)EXECUTION</u>

```
#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;

}
```

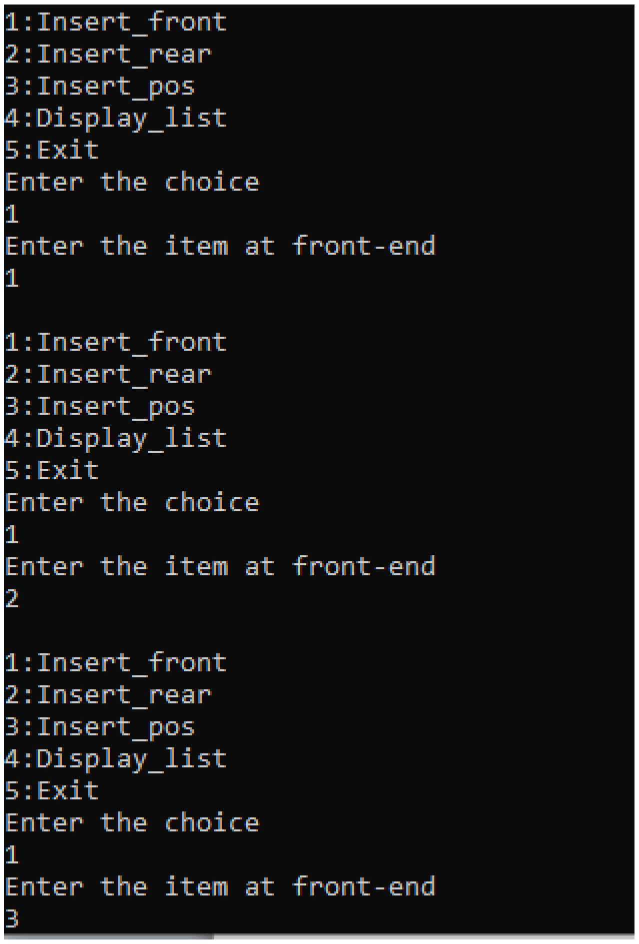
```
void display(NODE first)
{
    NODE temp;
    if (first == NULL)
        printf("list empty cannot display items\n");
    else
        printf("Contents of the list:\n");
    for (temp = first; temp != NULL; temp = temp->link)
    {
        printf("%d\n", temp->info);
    }
}
void main()
{
    int item, choice, pos;
    NODE first = NULL;

    for (;;)
```

```
   {

printf("\n1:Insert_front\n2:Insert_rear\n3:Insert_pos\n4:Displa
y_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;
        case 5:
           exit(0);
           break;
           default:printf("Invalid choice\n");

        }
    }
}
```

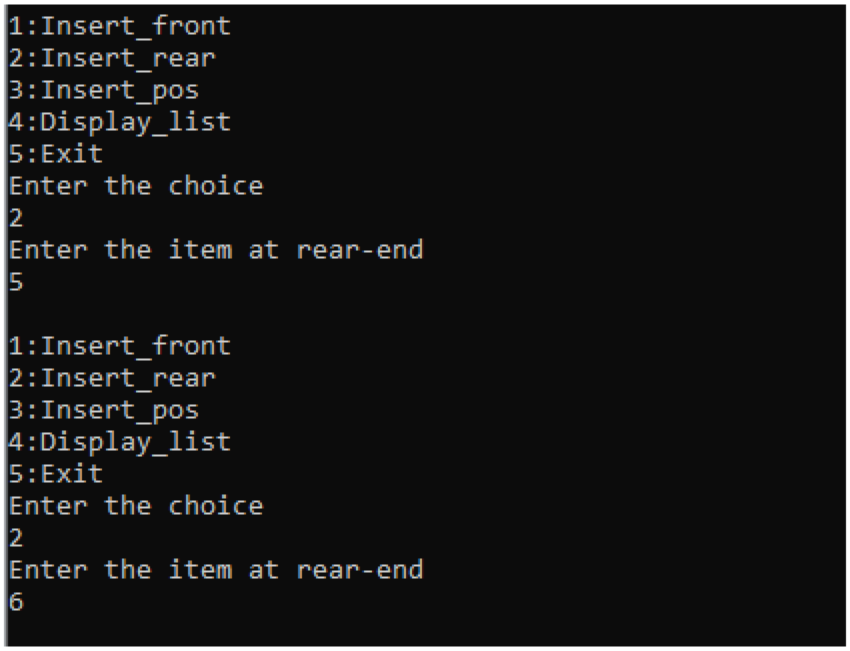
OUTPUT:

1.insert front

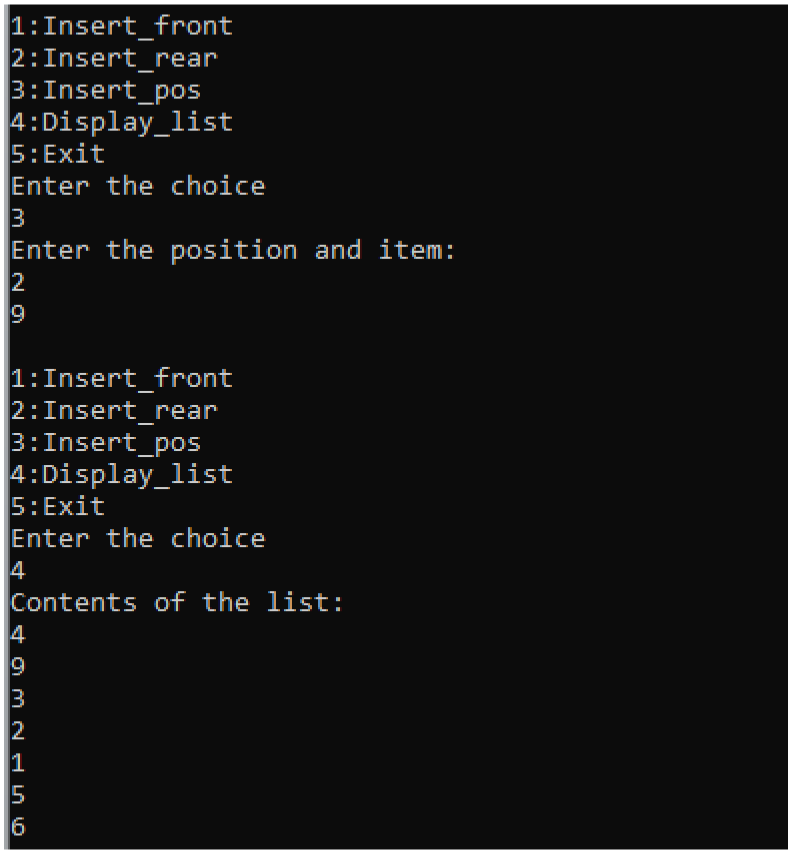


```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
1
Enter the item at front-end
1

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
1
Enter the item at front-end
2

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
1
Enter the item at front-end
3
```

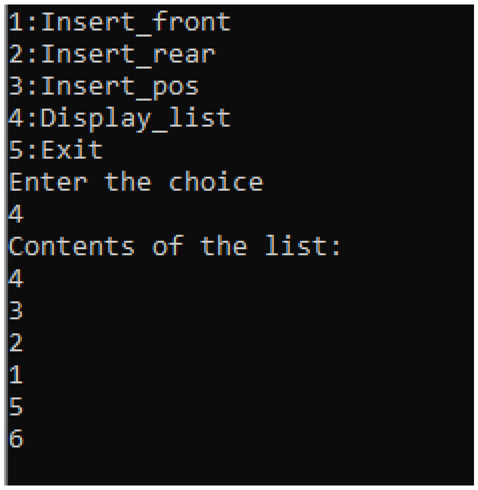
2.insert rear


```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
2
Enter the item at rear-end
5

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
2
Enter the item at rear-end
6
```

3.insert pos


```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
3
Enter the position and item:
2
9

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
4
Contents of the list:
4
9
3
2
1
5
6
```

4.display



```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
4
Contents of the list:
4
3
2
1
5
6
```