

Write a program to implement singly linked list with the following operations:

a. Create a linked list

b. Insertion a node at first position at any position and at end of the list.

Display the contents of linked list.

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct node {
```

```
    int info;
```

```
    struct node *next;
```

```
};
```

```
struct node *createln() {
```

```
    struct node *p;
```

```
    struct node *start = NULL;
```

```
    int item;
```

```
    printf("Enter -999 to exit\n");
```

```
    scanf("%d", &item);
```

```
    while (item != -999) {
```

```
        p = (struct node *)malloc(sizeof(struct node));
```

```
        p->info = item;
```

```
        p->next = start;
```

```
        start = p;
```

```
        scanf("%d", &item);
```

```
}
```

```
    return start;
```

```
}
```

```
struct node *insertfirst(struct node *start, int item) {
```

```
    struct node *p = (struct node *)malloc(sizeof(struct node));
```

```
    p->info = item;
```

```
p->next = start;
return p;
}

struct node *insertatlast(struct node *start, int item) {
    struct node *p = (struct node *)malloc(sizeof(struct node));
    struct node *temp;
    p->info = item;
    p->next = NULL;

    if (start == NULL)
        return p;

    temp = start;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = p;
    return start;
}

struct node *insertatposition(struct node *start, int index, int item) {
    struct node *p = (struct node *)malloc(sizeof(struct node));
    struct node *temp = start;
    int i = 0;

    p->info = item;

    if (index == 0) {
        p->next = start;
        return p;
    }
```

```
}
```

```
while (i < index - 1 && temp != NULL) {
```

```
    temp = temp->next;
```

```
    i++;
```

```
}
```

```
if (temp == NULL) {
```

```
    printf("Invalid index\n");
```

```
    free(p);
```

```
    return start;
```

```
}
```

```
p->next = temp->next;
```

```
temp->next = p;
```

```
return start;
```

```
}
```

```
void display(struct node *start) {
```

```
    struct node *temp;
```

```
    if (start == NULL) {
```

```
        printf("Linked list is empty\n");
```

```
        return;
```

```
}
```

```
temp = start;
```

```
printf("Elements are:\n");
```

```
while (temp != NULL) {
```

```
    printf("%d\n", temp->info);
```

```
    temp = temp->next;
```

```
}
```

```
}
```

```
int main() {
    struct node *head = NULL;
    int choice, val, index;

    while (1) {
        printf("\n1) Create linked list\n2) Insert at first\n3) Insert at last\n4) Insert at position\n5)
Display\n6) Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                head = createlk();
                break;
            case 2:
                printf("Enter value to insert: ");
                scanf("%d", &val);
                head = insertfirst(head, val);
                break;
            case 3:
                printf("Enter value to insert: ");
                scanf("%d", &val);
                head = insertatlast(head, val);
                break;
            case 4:
                printf("Enter value to insert: ");
                scanf("%d", &val);
                printf("Enter index to insert: ");
                scanf("%d", &index);
```

```

head = insertatposition(head, index, val);

break;

case 5:

display(head);

break;

case 6:

printf("Exiting program.\n");

return 0;

default:

printf("Invalid choice.\n");

}

}

}

```

Output:

```

PS C:\Users\WISCHAL\OneDrive\Documents\Desktop\dsa> cd "c:\Users\WISCHAL\OneDrive\Documents\Desktop\dsa"
PS C:\Users\WISCHAL\OneDrive\Documents\Desktop\dsa> cd "c:\Users\WISCHAL\OneDrive\Documents\Desktop\dsa\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) {
.\tempCodeRunnerfile

1) Create linked list
2) Insert at first
3) Insert at last
4) Insert at position
5) Display
6) Exit
Enter your choice: 1
Enter -999 to exit
10
20
30
-999

1) Create linked list
2) Insert at first
3) Insert at last
4) Insert at position
5) Display
6) Exit
Enter your choice: 2
Enter value to insert: 40

1) Create linked list
2) Insert at first
3) Insert at last
4) Insert at position
5) Display
6) Exit
Enter your choice: 3
Enter value to insert: 0

1) Create linked list
2) Insert at first
3) Insert at last
4) Insert at position
5) Display
6) Exit

```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows files in the workspace, including `queue.c`, `circularQueue.c`, `sllInsertion.c`, `circularQueue.exe`, `circularQueue.pdf`, `Infix to postfix.pdf`, `infixToPostfix.c`, `infixToPostfix.exe`, `queue operations.pdf`, `queue.c`, `queue.exe`, `sllInsertion.c`, `sllInsertion.exe`, `stack.c`, `stack.exe`, `tempCodeRunnerFile.c`, `tempCodeRunnerFile.exe`, and a task titled "Write a program to st...".
- TERMINAL**: Displays the output of a C program. The program menu includes:
 - 3) Insert at last
 - 4) Insert at position
 - 5) Display
 - 6) ExitThe user enters choice 3, then "Enter value to insert: 0".
The program menu includes:
 - 1) Create linked list
 - 2) Insert at first
 - 3) Insert at last
 - 4) Insert at position
 - 5) Display
 - 6) ExitThe user enters choice 4, then "Enter value to insert: 100".
The program menu includes:
 - 1) Create linked list
 - 2) Insert at first
 - 3) Insert at last
 - 4) Insert at position
 - 5) Display
 - 6) ExitThe user enters choice 5, then "Elements are:
40
30
20
100
0".
The user then enters choice 6, then "Exiting program."

The terminal also shows the command prompt: PS C:\Users\NISCHAL\OneDrive\Documents\Desktop\dsa>