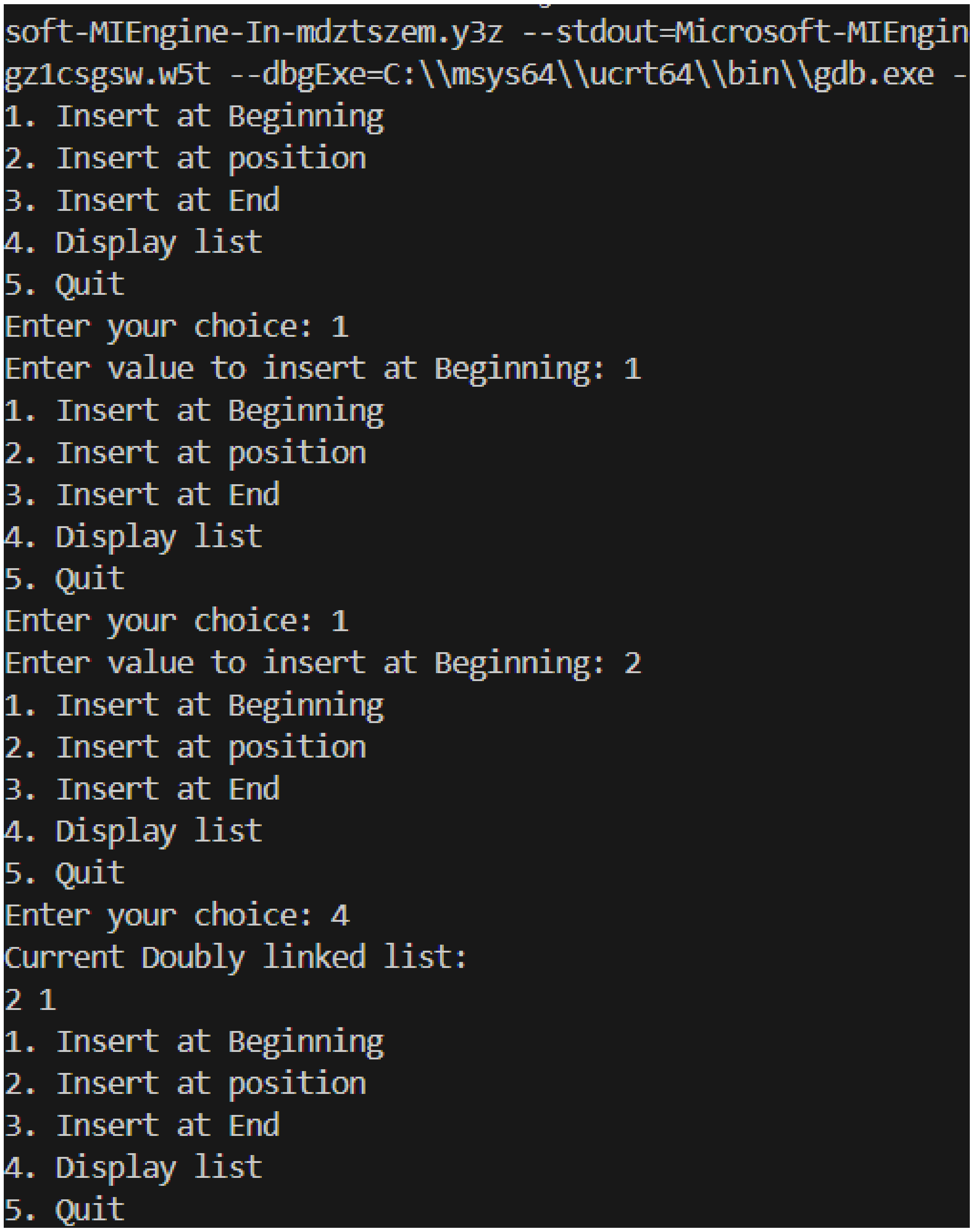
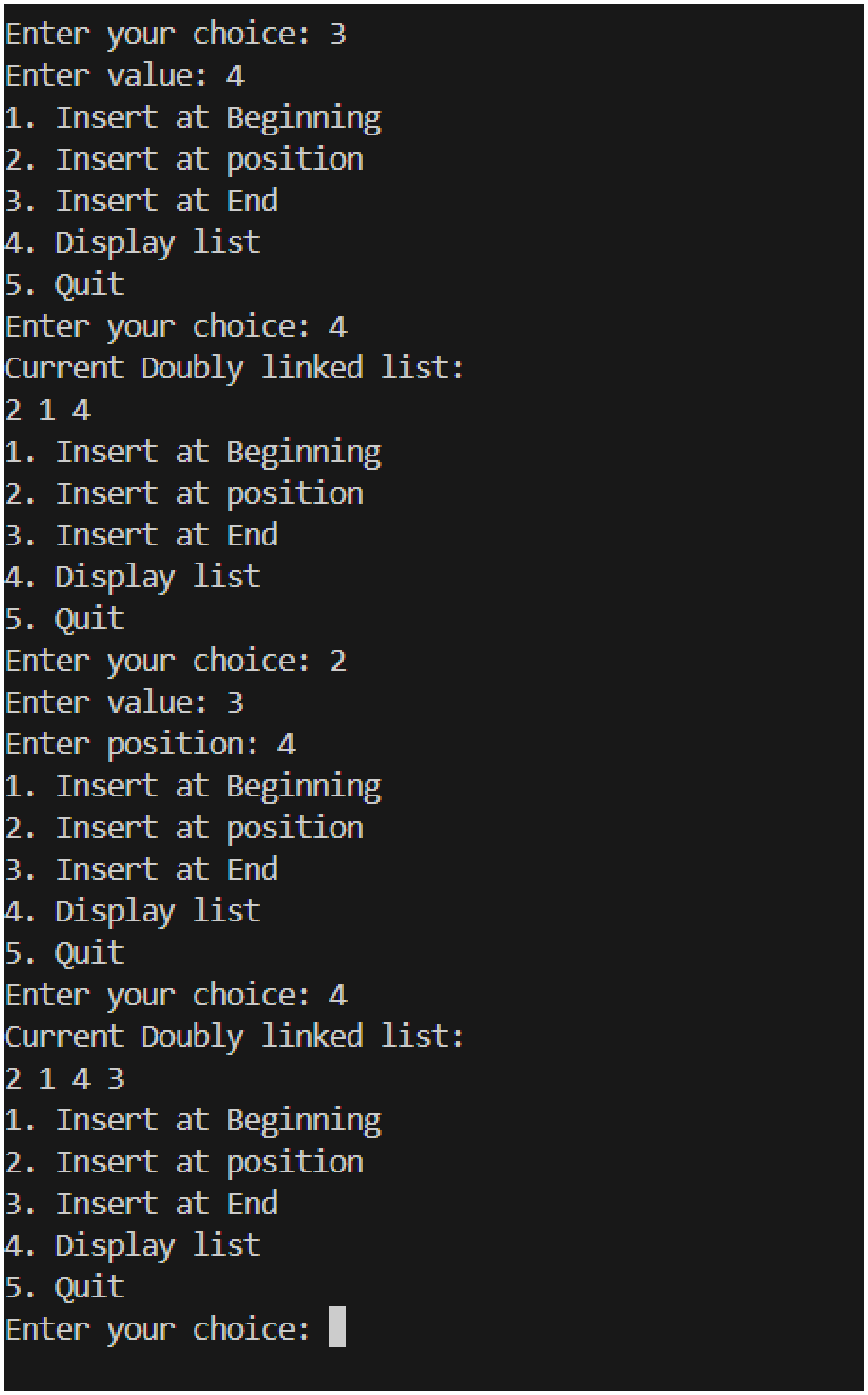
```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    newNode->next = *head;
    (*head)->prev = newNode;
    *head = newNode;
}

void insertAtPos(struct Node** head, int data, int pos) {
    struct Node* newNode = createNode(data);
    struct Node* temp = *head;
    int count = 1;

    if (pos <= 1) {
        insertAtBeginning(head, data);
        return;
    }

    while (temp != NULL && count < pos - 1) {
        temp = temp->next;
        count++;
    }

    if (temp == NULL) {
        printf("Position out of range\n");
        return;
    }

    newNode->next = temp->next;
    if (temp->next != NULL)
        temp->next->prev = newNode;
    temp->next = newNode;
    newNode->prev = temp;
}

void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }

    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }

    temp->next = newNode;
    newNode->prev = temp;
}

void displayList(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Node* head = NULL;
    int choice, value, position;

    while (1) {
        printf("1. Insert at Beginning \n");
        printf("2. Insert at position \n");
        printf("3. Insert at End \n");
        printf("4. Display list \n");
        printf("5. Quit \n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert at Beginning: ");
                scanf("%d", &value);
                insertAtBeginning(&head, value);
                break;

            case 2:
                printf("Enter value: ");
                scanf("%d", &value);
                printf("Enter position: ");
                scanf("%d", &position);
                insertAtPos(&head, value, position);
                break;

            case 3:
                printf("Enter value: ");
                scanf("%d", &value);
                insertAtEnd(&head, value);
                break;

            case 4:
                printf("Current Doubly linked list: \n");
                displayList(head);
                break;

            case 5:
                printf("Quitting program.\n");
                exit(0);

            default:
                printf("Invalid choice");
        }
    }
}
```

Output:



```
soft-MIEngine-In-mdztszem.y3z --stdout=Microsoft-MIEngin
gz1csgsw.w5t --dbgExe=C:\\msys64\\ucrt64\\bin\\gdb.exe -
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 1
Enter value to insert at Beginning: 1
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 1
Enter value to insert at Beginning: 2
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 4
Current Doubly linked list:
2 1
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
```


```
Enter your choice: 3
Enter value: 4
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 4
Current Doubly linked list:
2 1 4
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 2
Enter value: 3
Enter position: 4
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 4
Current Doubly linked list:
2 1 4 3
1. Insert at Beginning
2. Insert at position
3. Insert at End
4. Display list
5. Quit
Enter your choice: 
```

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

    struct Node
    {
        int data;
        struct Node* prev;
        struct Node* next;
    };

    struct Node* createNode (int data)
    {
        struct Node* newNode = (struct Node*) malloc (sizeof(struct Node
        newNode -> dadta = data;
        newNode -> prev = NULL;
        newNode -> next = NULL;
        return newNode;
    }

    void insertAtBeginning (struct Node** head, int data)
    {
        struct Node* newNode = createNode (data);
        if (* head = NULL)
        ~~{~~
            ~~* head = newNode~~
                ~~return;~~
        ~~}~~
        newNode -> next = * head;
        (*head) -> prev = newNode;
        * head = newNode;
    }
```

```
void insertAtPos(struct Node** head, int data, int pos)
{
    struct Node* newNode = createNode(data);
    struct Node* temp = *head;
    int count = 1;

    if(pos == 1)
     {
       insertAtBeginning(head, data);
       return;
     }
    while (temp != NULL && count < ~~position~~ pos -1)
    {
       temp = temp -> next;
       count ++;
    }
     if (temp == NULL)
      {
        printf("Position out of range \n");
        return;
      }
    newNode -> next = temp -> next;
    if (temp -> next != NULL)
        temp -> next -> prev = newNode;
    temp -> next = newNode;
    newNode->prev = temp;
}
```

```
void insert AtEnd (struct Node **head; int data)
{
    struct Node * newNode = createNode (data);
    if (* head = = NULL)
     {
          *head = newNode;
           return;
     }
     struct Node* temp = * head;
     while ( temp -> next ! = NULL)
      {
           temp = temp -> next;
      }
      temp -> next = newNode;
      newNode -> prev = temp;
}

void displayList (struct Node* head)
 {
     if (head = = NULL)
      {
          printf ("List is empty.\n");
          return;
      }
     struct Node* temp = head;
     while (temp ! = NULL)
      {
          printf ("%d ", temp->data);
          temp = temp -> next;
      }
     printf ("\n");
}
```

```
int main()
{
    struct Node* head = NULL;
    int choice, value, position;

    while (1)
    {
        printf("1. Insert at Beginning \n");
        printf("2. Insert at position \n");
        printf("3. Insert at End \n");
        printf("4. Display list \n");
        printf("5. Exit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);

        switch (~~ch~~ choice)
        {
            case 1 :
                printf("Enter value to insert at Beginning:");
                scanf(" %d", &value);
                insertAtBeginning (&head, value);
                break
            case 2 :
                printf("Enter value:");
                scanf("%d", &value);
                ~~insert~~ printf(" Enter position");
                scanf("%d", &position);
                insertAtPos (&head, value, position);
                break
            case 3 :
                printf("Enter value : ");
                scanf("%d", &value);
                insertAtEnd (&head, value);
                break;
```

PAGE :
DATE :

```
case 4:
    printf("Current Doubly Linked List: \n");
    displayList(head);
    break;

case 5:
    printf("Exiting program. \n");
    exit(0);

default:
    printf("Invalid choice");
  }
 }
 return 0;
}
```

O/p:

```
1. Insert at beginning
2. Insert at ~~End~~ Position
3. Insert at End
4. Display list
5. Exit
Enter your choice: 1
Enter value to insert: 1

Enter your choice: 1
Enter value to insert: 2
Enter your choice: 4
2, 1
Enter your choice: 3
Enter value to insert: 4
Enter your choice: 4
2, 1, 4
Enter your choice: 2
Enter value to insert: 3
Enter your choice: 4
2, 1, 3, 4
```

[illegible] 5/12/24